ViPNet Coordinator Linux

Software for building virtual private networks (VPN)

Administrator's Manual

Version 3.4-647

Infotecs GmbH
Hebbelstraße 41
D-14469 Potsdam

In Germany:
Tel: +49 (331) 817 03 76
Fax: +49 (331) 817 03 77
In England:
Tel: +44 (207) 871 75 67

Email: support@infotecs.biz
WWW: www.infotecs.biz

TABLE OF CONTENTS

INTRODUCTION

About this document

This document describes the purpose and use of the ViPNet Coordinator Linux software, the software required to build a ViPNet network, the main modes and features of the ViPNet software, as well as the full description of the program interface and possible settings.

The administrator's manual for the ViPNet software is organized in the following way:

- Chapter 1 (page 5) and its subsections contain general information about the purpose and use of the ViPNet software and its working modes.
- Chapter 2 (page 9) focuses on recommendations related to installing the ViPNet software.
- Chapter 7 (page 53) gives you the nuts and bolts of how to work with the ViPNet software.
- Chapter 8 (page 54) focuses on the Failover System, included in ViPNet Coordinator Linux and used to monitor system health and prevent disastrous failures, ViPNet control program and MFTP transport module as well. For more information refer to Failover System Administrator Guide.
- Chapter 9 (page 57) covers console utilities designed to view IP packets log as well as to obtain other information about ViPNet software operations.
- Chapter 10 (page 69) contains the description of some fine-tuning for ViPNet Coordinator Linux that can be used by qualified system administrators.

1 General Overview

1.1 About ViPNet Software

The ViPNet software is a universal software tool ensuring the reliable protection of a computer against unauthorized access to its various informational and hardware resources via the IP protocol when the computer works in local area networks or global networks, e.g. Internet.

The software is based on the driver that interacts directly with the network level driver, which makes the program independent of the operating system and its undocumented features. The driver controls the entire IP traffic, outgoing from the computer and incoming into the computer.

When the computer interacts with other computers with this software installed on them over the network, the Driver ensures establishing protected connections between such computers. During this process, the entire IP traffic between two computers is completely transformed (encrypted) according to the algorithm recommended by GOST (State Standard) 28147-89, which makes this traffic unavailable for other computers, including those having the same software installed on them. At the same time, all types of IP packets are encapsulated into a single type thus completely concealing the structure of information exchange. It is also possible to filter protected traffic according to the specified security policy. Traffic is encrypted using 256-bit keys.

1.2 Application Area

The ViPNet network traffic protection system allows you to solve various tasks related to the secure operation of computers in both local area networks and global networks:

- ViPNet makes it possible to establish protected connections between computers equipped with ViPNet over standard phone lines, in a LAN or over the Internet. It allows users to carry out negotiations and videoconferences protected against unauthorized access, send and receive any information. At the same time, not only a protected connection is established, but also external traffic is filtered to prevent access to the computer resources. It is enough to connect the computer to a phone outlet anywhere in the world or to a LAN that is connected to the Internet in order to establish such a protected connection.

- By installing ViPNet on the corporate web server and on computers that connect to it, you will get an informational system for corporate users completely protected against unauthorized access in the open Internet environment or in the local area network.

- When ViPNet is installed on workstations and LAN servers, information becomes securely protected against being intercepted and modified. It also becomes impossible to send forged information. It does not matter if unauthorized actions are carried out from the inside or from the outside in case the LAN is connected to an open global network, e.g. Internet.

When the ViPNet software is installed on database servers and workstations working in the terminal mode or using the «client-server» technology, both information protection and access restrictions are enabled for information resources on database servers.

The ViPNet Coordinator Linux software will work in networks based on the following link layer protocols:

- Ethernet
- PPP
- SLIP (including CSLIP, SLIP6, CSLIP6)

1.3 Approximate Topology of Corporate Network Based on the ViPNet Software

A corporate network consists of network nodes, on which either ViPNet Coordinator or ViPNet Client software is installed. Let’s call a network node with installed ViPNet Coordinator as a coordinator and a network with installed ViPNet Client – a client (a workstation instead of client is used as well). Coordinators are different because they can carry out a number of functions that clients do not have:

- Routing Server feature. Coordinator carries out routing of mail envelopes and control messages at interaction of network nodes between each other.

- IP Addresses Server feature. Coordinators store information about clients’ addresses. When some client is being switched on or off, it sends information about it to its coordinator (which is an IP Addresses Server for this client), while the coordinator sends this information to other clients so that all clients have information about each other's addresses. In case of several coordinators on the network, they exchange this information.

- Coordinators can work as proxy servers for clients. In this case other clients do not know the client’s real address, but only the address of its proxy server. A virtual address is used to connect to such a client.

- Coordinators can tunnel traffic from public network computers (for instance, from those on which it is impossible to install the ViPNet software for some reasons). In this

case traffic remains unprotected only on its way from the tunneled computer to the coordinator and becomes encrypted for the rest of its way.

- Open Internet Server feature. This feature allows you to place local network resources behind a specially configured coordinator thus allowing these resources to access Internet without physically disconnecting them from the local network.

Along with coordinators and clients, a corporate network includes the Network Control Center (NCC) and the Key and Certificate Authority (KCA). They perform administrative functions and are responsible for generating keys used for encryption, permitting/forbidding links between network objects, etc. They are not described in this manual, but it should be kept in mind that the ViPNet Coordinator Linux software works only if key sets created by NCC and KCA are available (for example created by the ViPNet Manager, which is a part of the ViPNet OFFICE software suite).

1.4 Software Package

A computer with the ViPNet Coordinator Linux software installed on it can be used as a ViPNet workstation or as an RS depending on what key sets are used on each particular computer. The ViPNet Coordinator Linux software includes the following functional parts:

- Low-level network protection driver iplir. It interacts directly with network card drivers and controls the entire traffic exchange of the local node with external network.
- Control daemon program. It loads the necessary parameters for the driver, sends and receives information about clients' IP addresses, logs traffic information, etc. It is recommended to keep this program always running, but when it is stopped, the driver goes on working and traffic exchange is not interrupted.
- Console utilities. They allow you to view information about ViPNet objects and traffic logs, change the configuration, etc.
- Failover daemon, which provides the functionality of Failover System. For more information, refer to 8 section.
- MFTP daemon, which provides the ViPNet business mail messages and files exchange, makes it possible to update host links and key sets on workstations remotely, as well as update the ViPNet software distribution package remotely.
- SNMP daemon that allows you to retrieve the ViPNet statistics from remote hosts via the SNMP protocol.

1.5 Main Security Levels

The kernel of the ViPNet software responsible for encrypting and filtering packets can work on several security levels. Each of these levels is a certain set of rules according to which encrypted and unencrypted packets are processed. This level is specified separately for each network interface, which allows you to customize the work according to each user's needs.

Note: The word *'Open'* means unencrypted.

Level 1: Encryption is active. All open IP traffic is blocked. It is the strictest and most secure level**.** This level allows you to work only with computers that also have ViPNet installed on them and you will not be able to work with unprotected computers since the system passes only protected IP packets and blocks any other packets.

Level 2: Encryption is active. Registered traffic is allowed. On this level the system will accept all protected connections, as well as packets for which the corresponding permission is specified in filters for the public network and block any other packets. Any incoming and outgoing connections that are not explicitly specified in connection filters are completely blocked.

- **Level 3: Encryption is active. The «boomerang» mode is used for open traffic.** This level is used to ensure more secure work over the Internet with open servers and other computers. On this level the driver is configured to work only with that remote unprotected computer, and only with that application, and only via that protocol which the user thinks it is possible to work with. It means that the driver allows connections **initiated by your computer**. Any incoming packets of unknown types are blocked in this case. The driver automatically registers and passes any outgoing packets for some time. After such a packet is registered, all incoming and outgoing packets of the registered type are accepted and processed. All packets registered in filters for the public network are passed as usual no matter who initializes a connection. When you use this level to work in the LAN with open computers in it, you can access such computers while it is impossible to access your computer from them.

Level 4: Encryption is active. All open IP traffic is allowed. You can access protected servers or connect to protected computers, but your computer will be completely open for unprotected connections. It is recommended to use this level only for test and configuration purposes.

Level 5: Encryption is disabled. All open traffic is allowed. This level completely disables the ViPNet driver for this interface. In this case protected packets are not decrypted and it is impossible to connect to protected workstations. It is not recommended to use this mode except for special cases, for example, when organizing the control channel in the hot failover system. As a rule, such cases are particularly singled out in the documentation.

The ViPNet software is mostly a self-configuring system. In the process of its operation the ViPNet control program sends special protected packets that allow computers with this software see each other in the LAN, inform each other about their addresses if they change. For computers to see each other in a wide area network, it must have several computers working as IP addresses servers.

1.6 Firewall Modes for the ViPNet Software

For the nodes to be able to work from any network points, the ViPNet software can work in four main modes:

1. Autonomous work without using an external firewall (the firewall type is ***"No external firewall"***).
2. Work via a ViPNet coordinator (the firewall type is "***ViPNet coordinator"***)
3. Work via an external firewall (device) with address translation (NAT) where it is possible to set up static address translation rules (the firewall type is "***Static NAT"***)
4. Work via an external firewall (device) with address translation (NAT) where it is difficult or impossible to set up static address translation rules (the firewall type is "***Dynamic NAT"***)

The mode for each node is selected depending on how particularly the notification and registration of node statuses are implemented in the ViPNet network and their location of nodes on the network:

- once connected to the network, each node (if node is client) sends to its coordinator (its IP addresses server) or other coordinators (if node is coordinator) the necessary information about its addresses and ways to access it,

- once connected to the network and during its operation, each node (if node is client) receives from its coordinator (IP addresses server) or other coordinators (if node is coordinator) the necessary information about the addresses of other nodes connected to it and ways to access these addresses.

So, if a node has an IP address available to any other nodes with which it is supposed to interact according to the general rules of packet routing in the IP network (for instance, it has a real Internet or corporate network IP address), this node can just send other nodes only its IP address. The ***"No external firewall"*** mode can be selected in this case. This mode should be selected for a coordinator for it to be able to act as a proxy server for other ViPNet workstations, i.e. for other workstations to be able to work in the "***ViPNet coordinator"*** mode using this node as a ViPNet proxy server.

If a node has a private IP address that cannot be used by some other nodes to access it according to the general routing rules, that is, there is a firewall or another device set up at the border of the LAN and it works as network address translation (NAT), other nodes have to know much more information about this node. To provide an uninterrupted access to this node, not only information about the address of this node can be required, but also information about the addresses and current access ports through NAT device at the moment can be required as well. In this case you should select one of the Firewall modes for this node.

The "***ViPNet coordinator"*** mode is selected for nodes if the network has a ViPNet coordinator that can work autonomously or via other NAT devices.

The "***Static NAT"*** mode is selected for nodes if there is no ViPNet coordinator in the LAN, but there is a firewall that can do NAT and that makes it possible to specify static NAT rules ensuring work with a certain internal address via the UDP protocol using the specified port.

The "***Dynamic NAT"*** mode is selected for nodes in case there is no ViPNet coordinator on the LAN and connections to the network are established via a certain NAT device where it is difficult to specify static NAT rules. It should be mentioned that this mode is the most universal one and a node working in this mode will be able to work in case of other connection methods as well.

It also should be mentioned that if nodes are in one LAN and are available for each other within the area of broadcast packets, they can always interact with each other directly by their IP addresses no matter what mode is selected for them.

You can find the detailed description of how to configure each mode in section 6.5.

Regardless of the modes described in this section, it is possible to switch each network interface to any security level described in section 1.5.

2 Installing ViPNet Linux

To install ViPNet Linux software, you need a computer with Pentium-III processor 450MHz or higher, not less than 300 MB of free space on the hard disk, not less than 128 MB of RAM. The computer must have the 32-bit Linux operating system of one of the following distribution packages installed on it:

- Linux XP 2008 Server;
- Linux XP 2008 Desktop Secure Edition;
- RedHat Enterprise Linux 4.0 AS;
- Open SuSe Linux 11.1;
- SuSe Linux 10.0;
- SuSE Linux Enterprise Server 10;
- SuSe Linux Enterprise Server 10 SP1, SP2;
- Slackware Linux 10.2 (2.4.31 kernel only);
- Slackware Linux 12.0 (2.6.16.52 kernel only from ftp://kernel.org);

- Ubuntu 8.04 LTS Desktop;
- Debian Etch 4.0 r1.

ViPNet Linux may work with other Linux distribution packages, but it is not guaranteed. No 64-bit Linux version is supported yet. Also, it is not guaranteed that the ViPNet Linux software will work on a computer with third-party traffic transformation tools, third-party protection tools working on the channel, network and transport layer (in particular, ipchains and iptables) and third-party traffic encryption tools (IPSec, etc.).

The system should use a Linux kernel from a subset of 2.4.x versions (2.4.2 to 2.4.31 inclusive) or 2.6.x versions (up to 2.6.29 inclusive). When using a kernel patch as a network packet interception method for the ViPNet driver (see section 2.1), compatibility with Linux kernel sources supplied with distribution kits is not supported. In this case we only guarantee compatibility with official versions of Linux kernels from the ftp.kernel.org FTP server (see section 2.1).

Root access is necessary to install the software (user ID = 0). During the installation you will need to have access to the console of the computer where you are installing ViPNet since the computer may become unavailable via the network until you specify all the necessary parameters.

In order to install the software, you need the following:

- ViPNet Linux software distribution package;
- The distribution key set that will be used on this computer. You have to obtain this key set from your ViPNet network administrator. The administrator must also tell you your password that will be used to access your key set.

ViPNet Linux extensively interacts with other ViPNet components (NCC, KC) running under Windows. During this process some files with keys and other information are sent to Linux via the remote update system. Since in Windows (unlike in Linux) the names of files and directories are not case-sensitive, the actual file name in Windows may contain lowercase and uppercase letters, which is determined by a lot of factors not always depending on the program that creates them. That is why ViPNet Linux converts all names of files and directories it uses to lowercase. In order to avoid possible problems, you should observe one simple rule: **the names of all directories where you unpack the distribution package of ViPNet Linux, where the key set is stored, etc. must contain only Latin lowercase letters and the symbols «minus», «underscore», etc. Do not use uppercase letters in their names**.

Before installing ViPNet, you should check if all necessary system components are installed and install those that are missing. The ViPNet installation script does not check if these components are installed in the system and if you try to install ViPNet on a computer where they are not installed, the system may stop functioning.

The software requires the following components to be installed (Table 1):

Table 1

Package name	**Version not earlier than**	**Version not later than**
Linux kernel 2.4.x, or Linux kernel 2.6.x	2.4.2 2.6.0	2.4.31 2.6.24
bash – console interpreter	1.10	
sysklogd – system logfile manager	1.3	
modutils – library for working with modules	2.1	
sh-utils – library for the console interpreter	2.0	

fileutils – library for working with files	4.0	
Psmisc – library for working with processes	18.3	
net-tools – network tools	1.53	
grep – line analysis tool	2.3	
logrotate- logs rotation utility	3.0	
cron – task scheduler (at system startup, an automatic startup is necessary for work of log rotation function, see chapter 13)	3.0	
ethtool – the network interface parameter configuration utility	1.3	

Along with the above packages, the software requires the following packages for its installation (they can be deleted after ViPNet is successfully installed):

Package name	**Version not earlier than**	**Version not later than**
gcc – C compiler	3.x series not earlier than 3.2 4.x series not earlier than 4.0.2	
build-essential (for Ubuntu 8.04 LTS Desktop)	11.3	
glibc-2.3.5, glibc -2.3.5-profile (for Lunux Slackware 10.2)	2.3.5	
At using kernel patch to intercept network packets (see section 2.1): Linux kernel header files (Linux kernel source code if using patch technology) 2.4.x or Linux kernel header files (Linux kernel source code if using patch technology) 2.6.x	2.4.2 2.6.0	 2.4.31 2.6.24
tar – archiver	1.13	
make – build script processor	3.75	
gzip – file compressor	1.2	
patch – patch installation tool	2.5	

To properly install ViPNet software you need bash command shell. If bash is not a default command shell on your Linux system (for example, on Ubuntu 8.04 systems dash is a default shell) you should set bash as a command shell. To check the current command shell type ls –l /bin/sh, where symbolic link points to the current command shell. To change command shell, if using Ubuntu 8.04, execute dpkg-reconfigure dash command. When prompted, you should say no about using dash and choose bash as a command shell.

ViPNet Linux currently supports 2 methods of intercepting network packets: using a **kernel patch** and using the **NETFILTER** technology. The first method requires a special kernel patch that comes together with the ViPNet Linux distribution package to be installed before the installation of ViPNet Linux. After you install the patch, you must rebuild the Linux kernel and start installing ViPNet Linux only after that. Besides, when using this method, compatibility with Linux kernel source code supplied in distribution package is not supported. With the **NETFILTER** technology, there is no need to install the patch, the packet interception feature is a

standard component of any supported Linux kernels. In most cases the necessary parameters are specified by default and there is no need to rebuild the kernel. If you select this packet interception method, you can use kernels coming together with the supported Linux distribution packages and not only the official versions of the kernels.

You should choose the method for ViPNet Linux integration yourself. **In case both methods described above are available, ViPNet Linux will use the kernel patch**. If you need to change the packet interception method later on, you will have to activate the necessary method and reinstall ViPNet Linux.

2.1 Using the Kernel Patch

When using this method network packet interception method, compatibility with Linux kernel source code included in distribution kits is not supported. Distribution kit creators make changes to them, frequently insufficiently considered, which lead to incorrect building and operation of kernel modules that were built for working with a standard kernel. We guarantee compatibility, and recommend using only the official version of the Linux kernel from the ftp.kernel.org FTP server (the last kernel of the required series), rebuilding it according to your own needs, and installing ViPNet on it. You should also remember, that you must rebuilt the kernel and install ViPNet using the same compiler version.

To install the patch, unpack the ViPNet Linux distribution package into any directory using the command tar xvfz <file>. It will create the distribute directory with the patch subdirectory in it where patches for various version of the Linux kernel are stored. Make sure that the source code of your kernel version is located in the directory /usr/src/linux. Then copy the patch file for your version of the Linux kernel from the distribute/patch directory into the directory /usr/src. Kernels 2.4.x and 2.6.x series are currently supported. You should use the file iplir.patch.2.4.x for versions 2.4.x up to 2.4.19 inclusive, iplir.patch.2.4.20. For kernel versions 2.6.x to 2.6.12 inclusive, you should use the iplir.patch.2.6.x file, and for versions 2.6.13 and above you should use the iplir.patch.2.6.13 file. Then go to the /usr/src directory, and install the patch using the patch –p0 <[patch file name] command. If the patch utility reports errors during the installation of the patch, it means that the Linux kernel on your computer does not belong to 2.4.x and 2.6.x series, or it has some nonstandard modifications, or you have selected the wrong patch file. In this case you should either install standard Linux kernel 2.4.x or 2.6.x series and select the correct file the way it is described above or select the other packet interception method.

If the patch has been installed correctly, you should go to the directory /usr/src/linux and start the kernel configurator using either the command make menuconfig for the text mode or the command make xconfig for the graphical mode. You should set the following options in the configurator:

- ENABLE the option IpLir Crypting network driver in the Networking options section;
- ENABLE the option Loadable module support in the Loadable module support section. We also recommend that you enable the option Set version information on all symbols in module, which allows you to avoid possible problems next time the kernel is updated. Starting from version 2.8-140, it is not necessary to enable this option, but we recommend that you enable it if you are not a professional Linux administrator and do not quite understand what you need it for. **This option is absent for kernels 2.6.x series and there is no need to enable it.**

  The rest of options are set at the administrator's discretion and according to the common sense.

You should build the kernel after you exit the configurator. To do it for kernels of 2.4.x series, execute the following commands in the following sequence:

make dep
make bzImage
make modules
make install
make modules_install

For kernels 2.6.x series, execute the following commands in the following sequence:
make
make install
make modules_install

Make sure that the kernel that has been just built will be loaded after you restart the system. To do it, make the necessary changes in the file /etc/lilo.conf and in other files if necessary. Restart the system. You can use the following command to check if the ViPNet patch is present in the loaded kernel 2.4.x series:

ksyms –a | grep set_packet_filter | wc –l

If the command returns 1 or a greater number, everything is installed correctly, if it returns 0, the patch has not been installed correctly or a kernel other than the rebuilt one has been loaded.

You can use the following command to check if the patch is installed for kernels 2.6.x series.

cat /proc/kallsyms | grep set_packet_filter | wc –l,

or the following command if the file /proc/kallsyms is not available:

cat /boot/System.map–`uname -r` |grep set_packet_filter | wc –l

If the command returns 1 or a greater number, everything is installed correctly, if it returns 0, the patch has not been installed correctly or a kernel other than the rebuilt one has been loaded.

2.2 Using the NETFILTER Technology

If you use this method of intercepting network packets, you can use either kernels coming together with your distribution package or the official versions of the Linux kernels.

In most case when kernels coming together with Linux distribution packages are used, kernel parameters necessary for using the **NETFILTER** technology are already set by default and there is no need to rebuild the kernel. To check if **NETFILTER** works properly, you should execute the following command:

For kernels 2.4.x series

ksyms –a | grep nf_reinject | wc –l

For kernels 2.6.x series

cat /proc/kallsyms | grep nf_reinject | wc –l,

or if the file /proc/kallsyms is not available

cat /boot/System.map–`uname -r` |grep nf_reinject | wc –l

If the command returns 1 or a greater number, **NETFILTER** is enabled and **there is no need to rebuild the kernel**. If it returns 0, you should reconfigure the kernel with the support of **NETFILTER** and rebuild it.

To enable the support of **NETFILTER**, you should restore the source code of the kernel being used. We recommend using the official version of the Linux kernel from the FTP server ftp.kernel.org because the kernel is not always correctly built if you use the source code coming

together with Linux distribution packages. Then you should go to the directory with the source code of the kernel and run the kernel configurator using either the command make menuconfig for the text mode or the command make xconfig for the graphical mode. You should set the following options in the configurator:

- ENABLE the option Network packet filtering in the Networking options section;
- ENABLE the option IP tables support in the Netfilter Configuration section;

You should build the kernel after you exit the configurator. To do it for kernels 2.4.x series, execute the following commands in the following sequence:

```
make dep
make bzImage
make modules
make install
make modules_install
```

For kernels 2.6.x series, execute the following commands in the following sequence:

```
make
make install
make modules_install
```

Make sure that the kernel that has been just built will be loaded after you restart the system. To do it, make the necessary changes in the file /etc/lilo.conf and in other files if necessary. Restart the system. Check if **NETFILTER** works properly using the method described above.

Important note. When the **NETFILTER** technology is used with some Linux distribution packages **without rebuilding the kernel**, it is necessary to install a number of packages needed to build ViPNet drivers.Below you can see the list of packages (Table 2).

Table 2

Distribution package name	Version
SuSe Linux 10.0	kernel-source
Slackware 10.2	kernel-source-2.4.31-noarch-1
SuSE Linux Enterprise Server 10.0	kernel-source-2.6.16.21-08
Debian Etch 4.0	linux-headers-2.6.18-5-686

2.3 Installing ViPNet Linux

To run the installation program, go to the distribute directory where the unpacked ViPNet Linux distribution package is located and execute the command install.sh. The installation software will build drivers, install them in the system, install applications and configure ViPNet Linux. The installation process consists of the following steps:

1. Selecting the ViPNet Linux distribution package. In this step the installation software searches the current directory for ViPNet distribution packages available for installation. The current directory usually contains only one distribution package named distribute.tar.gz, which the installation software will offer you to install. In some specific

cases you may need to choose another distribution package. In this case you should specify the directory that should be searched for the distribution package, search again and select the necessary distribution package.

2. Unpacking the selected distribution package. After you successfully select the distribution package, the installation software unpacks it into a temporary directory and then displays the license agreement. Moving to the next installation step is possible only after the user accepts the license agreement.
3. The analysis of the installed ViPNet configuration. In the current step the installation software searches for installed ViPNet configurations. If such a configuration is detected, the user is asked if he would like to use the current configuration. If the answer is positive, the configuration is preserved. If the answer is negative, the user will be offered to select a new configuration during the installation process.
4. Preparing for installation. This stage is used to perform several additional checks and other actions required for further installation (determine the packet interception method, Linux distribution kit support, compiler, check available space on the hard driver, etc.).

   This stage is used to analyze the availability of packages required for installation (see. Table 1). In case a required package is not found in the system, the installer aborts, displaying a respective message.
5. Moving to the next installation step is possible only if the preparation step is successfully completed.

   If an installation is being done on a computer with installed ViPNet software and a version of software you are installing is less than a version of installed software, that on this step, the setup program will search rollback configuration with lower or the same version of the software you are installing (see chapter 5). After that the setup program will inform a user that installation over more new version is being done and, probably, configurations files won't be accepted by version of the software you are installing without editing manually. If an appropriate rollback configuration is found, the setup program will inform that you will be able to restore the appropriate files from data base and then the setup program will output the configuration name, date of backup and software version number. Then you will be offered to choose one of three variants:

   a. use current configuration files
   b. restore configurations files from data base (this variant will be offered if an appropriate configuration is found)
   c. abort the installation

   In most cases, for correct work of the software it's recommended to restore configurations files from data base if an appropriate configuration was found.

6. Installing drivers. Drivers for the current Linux kernel are completely built and installed.
7. Installing applications. In this step, applications, scripts, configuration files, startup scripts are installed.
8. Saving the parameters for the automatic launch of applications.

In case an existing installed ViPNet Linux configuration was detected in step 3 and the user decided to keep it, the user will be offered to start ViPNet services after step 7 is completed. This is the end of the installation process. If the user refuses to start the services, he must either start them manually or restart the system.

9. Selecting a ViPNet Linux configuration file (a distribution key set). In this step the installation software searches the current directory for available configuration files (files with the .dst extension). The user is also offered to select another directory for searching or finish the installation. If the installation is finished in this step, the user will have to select the configuration file manually and unpack it into the selected directory using the command /sbin/unmerge <file.dst> <directory> and then configure ViPNet (see section 2.4).
10. Selecting a directory for installing the selected configuration. If an existing installed configuration was detected in step 3, the user will be offered to overwrite it with the new configuration in the same directory. If the user refuses to do it or if no installed configuration was detected, the user will be offered to select the directory for installing the configuration. After the directory is selected, the configuration file is automatically unpacked into this directory. In case the hard drive partition used to unpack the distribution key set has less free space then required, an error message is displayed, and installation is aborted.
11. User authentication. To finish the installation successfully, the user should enter the correct user password for the selected ViPNet configuration. After the password is entered, it is verified and saved if it is correct. If the password check fails, the user is offered to enter the password again and so on. If the user refuses to enter the password, the installation is finished incorrectly. In this case the user should configure ViPNet Linux manually for it to work correctly (see section 2.4).

After the last installation step is successfully finished, the user will be offered to start ViPNet Linux services. This is the end of the installation process. If the user refuses to start the services, he must either start them manually or restart the system.

The installation process is logged to the file /var/log/vipnet_install.log. Besides, the installation software creates the file /var/log/vipnet_files.txt containing paths to all installed files. In case the installation process is successfully finished, both files are copied into the directory where the ViPNet configuration is installed.

2.4 Manual ViPNet Linux Configuration

The user can interrupt the work of the installation software on any of the stages described above. Besides, it may turn out to be necessary to change the configuration after the installation. In these cases ViPNet Linux should be configured manually. The manual configuration means unpacking the distribution key set (if it is not unpacked yet) and setting environment parameters.

A distribution key set is a file with the .dst extension. Select (or create if necessary) the directory where the key set will be stored and unpack it into this directory using the command /sbin/unmerge <file.dst> <directory>. Then edit the file /etc/iplirpsw that stores the location of the key set. It is a text file with the following structure: the first line is the full path to the directory where the key set is located, while the second line is the password for accessing this set. The rest of its lines are ignored. After the installation this file contains two sample lines showing its syntax. You should delete these lines and insert your own ones. Also, edit the file /etc/iplirnetpsw, its first line should contain the password, while the second line should be left unchanged with 127.0.0.1. You can find a more detailed description of the configuration file /etc/iplirnetpsw in the manual dealing with the use of console utilities.

Beginning from version 2.8-140, it is possible to enter the password from the keyboard. To do it, you should specify the word Keyboard or KeyLock instead of the password in the file /etc/iplirpsw. Then, in the process of launching the control program, a request to enter the password is displayed on the terminal (or on the system console if it happens when the computer is just being started) and the user can use the keyboard to enter it. If the entered password is

incorrect, the program offers to enter it once again. The number of attempts is not limited. If Keyboard is specified, the user can interrupt the process of entering the password by pressing Ctrl+C. It will prevent the control program from launching and the work with the network is blocked. If KeyLock is specified, it is impossible to interrupt the process of entering the password and if the user does not enter the correct password when the system is started, he will not be able to start the computer.

After you edit these main configuration files, you should make the setup launch of the control program that will create its configuration files. All operations with the control program are carried out with the help of the iplir script located in the /sbin directory. It will be useful to add this directory to PATH if it is not there yet, it will allow you to execute the control script using the command iplir. If the /sbin directory is not in PATH, the iplir script is executed with the help of the command /sbin/iplir. Some command shells (for example, tcsh) cache the contents of the directories in PATH, that is why the command iplir may fail to work right after the installation despite that the /sbin directory is in PATH. In this case you should execute the command that refreshes the cache in this shell (for example, rehash for tcsh). You can find more details about it in the manual to the command shell you use.

So, the setup launch of the control program is made with the command iplir check. After that the control program does not remain in the memory, but just creates its files and finishes. The created files are located in the user subdirectory of the directory where the key set is stored. The control program creates one main file named iplir.conf and one or several configuration files for network interfaces. If ViPNet Client is used, one configuration file named iplir.conf-ethall is used for all network interfaces. If ViPNet Coordinator is used, one configuration files is created for each network interface existing on the computer. These files are named iplir.conf-eth0, iplir.conf-eth1, etc.

The file iplir.conf contains parameters common for all interfaces and related to the private network. Its main content is the list of protected network nodes the computer can connect to and the settings of the local network node. This set of network nodes is defined by the network administrator and the user cannot add some protected computer to this list or remove a computer from it. But you can edit the settings (available for editing) for each node on the private network – its addresses, filters that will be used to connect to it, etc.

The files iplir.conf-<interface> contain information specific for each interface. It includes security levels and traffic log settings.

Additionally, after the iplir check command is executed, the open packet processing rules configuration file is created: firewall.conf. For more detail on working with the open network configuration see section 6.3.

Before you start working, you should look through the configuration files and edit them so that the information in them corresponds to the actual information. While creating a distribution key set, the administrator specifies IP addresses and other settings for each VPN host, but this information can be invalid or out of date due to the administrator's neglect. If a client is being configured, the IP address of its IP Addresses Server should be specified correctly. If a server is being configured and it is not the only one on the network, the addresses of all other servers it is allowed to connect to should be specified correctly. A more detailed description of how to edit configuration files is available below.

Then you should start the ViPNet Linux control program using the command iplir start. This command also loads the ViPNet driver if it is not loaded yet. The program is launched in the background mode. You can use the following command to check if it is in the memory: ps afx |grep –v grep |grep iplircfg |wc –l. If this command returns a number greater than 0, the control program is running.

The launch of the control program may fail due to a number of reasons: a corrupted distribution package, an incorrect password specified in the file /etc/iplirpsw, etc. Once started and

later in its work, the control program sends information and error messages to syslog using the facility 'daemon' and the level 'err' for errors or 'info' for information messages. Besides, during the startup, error messages are also sent to the console. Usually the system log is stored in the file /var/log/messages.

If the program has been started, you can check the work of the private network, for example, with the help of the command ping by sending packets to some protected network node the address of which is sure to be correct in the file iplir.conf. If packets are received in respond, the network functions normally. If you cannot connect to some protected host, you should edit the configuration files as described below.

If everything works, you should restart the system and make sure ViPNet is launched at startup. At successful installation, the following software will be launched at startup: the ViPNet drivers, the failover daemon (see 8 section) which lunches the controlling program (iplircfg daemon) and ViPNet MFTP transport module daemon (mftpd). After that, the system is completely ready to work in the private network without any additional actions from the user.

To enable the support of SNMP (viewing ViPNet statistics remotely), you should do the following:

- If you already have some SNMP daemon installed, delete it.
- Go to the distribute directory where the unpacked ViPNet Linux distribution package is stored and unpack the software archive file using the following command:

  tar xzvf distribute.tar.gz
- Run the install-snmp script from the snmp subdirectory of the directory with the ViPNet distribution package. You must have the root permissions to be able to do it. This script builds the SNMP daemon in such a way that it has ViPNet statistics available in it and installs it into /usr/local/sbin.
- Run /usr/local/sbin/snmpd. If you want to launch the snmpd daemon each time when the system is started, you should configure it manually because the installation script does not do it.
- If you want to receive snmp-traps, add the following lines to the configuration file snmpd.conf:

  informsink <the name of the receiving host>

  trapsink <the name of the receiving host>

  trap2sink <the name of the receiving host>

  where you should specify the actual name or address instead of <the name of the receiving host>.
- Import the file VIPNET-MIB.txt from this folder into your MIB browser. You can read how to do it in the documentation to your MIB browser. Information about ViPNet will be available in the following SNMP branch:
  .iso.org.dod.internet.private.enterprises.infotecs.vipnet.

See additional information about changes made in the system during the installation of ViPNet Linux in Appendix C. Changes made in the system when ViPNet Linux is installed and uninstalled.

Important note

If you decide to rebuild the Linux kernel to enable or disable some important options (such as Symmetric Multi-Processing Support) after the installation of ViPNet, ViPNet will not work after the new kernel is built and you will have to reinstall it. Moreover, if you try to load the

ViPNet driver built before to the new kernel, it will eventually lead to kernel panic. The approximate algorithm of rebuilding the kernel in such cases is as follows:

- Save the files of ViPNet configuration /etc/iplirpsw, /etc/iplirnetpsw, /etc/failover.ini, /etc/iplirSKnums
- Run the script /sbin/rmiplir, which will remove ViPNet from the system
- Restart the system
- Rebuild the kernel
- Restart with the new kernel
- Install ViPNet from its distribution package.
- Copy the saved files to the /etc directory
- Restart with ViPNet

Besides, when you are rebuilding the kernel, you should keep in mind that when some important options are changed (such as Symmetric Multi-Processing Support), it is **NOT ENOUGH** just to change the corresponding setting in the kernel configuration menu and go through the sequence of commands mentioned above to build the kernel. In most cases it will result in a non-functional kernel. Even the command make clean is not enough to build the kernel properly. In this case the approximate algorithm of building the kernel is as follows:

- Save the file .config, which contains the kernel configuration
- Execute the command make mrproper
- Copy the saved file .config back to its place
- Execute the command make menuconfig and make the necessary changes
- Execute the above sequence of commands to build the kernel

You can find more details about this in documentation to the Linux kernel.

3 Removing ViPNet Linux

If you need to completely remove ViPNet from the computer, you should use the script rmiplir, which is saved to the /sbin directory after the installation. You must run this script with the YES parameter (i.e. rmiplir YES), which excludes the possibility of removing ViPNet accidentally.

The script rmiplir removes the ViPNet software from the computer, its configuration files from the /etc directory, its device files from the /dev directory, etc. It does not remove the ViPNet key set. If you need to remove it, you can do it manually using the command rm. Besides, the rmiplir script is not removed as well. You should manually remove it from the /sbin directory.

After removing ViPNet, you should restart the system.

4 Updating Version of the ViPNet Linux

If you need to update the installed version of the ViPNet Linux software, you should do the following:

- Copy the distribution package of the new version to the computer and unpack it as described above.
- Go to the distribute directory where the unpacked ViPNet Linux distribution package is stored and execute the command install.sh (see section 2.3)
- If you are prompted to stop running ViPNet Linux services automatically, do it.

- In the process of installation, you will be prompted to save the current configuration during the installation process. You should give a positive answer to this question.
- After the installation is completed, you will be prompted to start ViPNet Linux services automatically. If you want the services to be started right after the installation, you should give a positive answer.

ViPNet Linux Software update during the working of the failover system included in the server hot failover cluster, is described in the "Server Failover System" administrator guide.

5 ViPNet Linux Software Backups (Returning to the Previous ViPNet Linux Software Version)

If after installing a new ViPNet Linux version over the old version some problems take place, you will need to return to the previous version of the software. When you install a new version over the old one, problems, related to parameters of configuration files, can arise. That means that old versions don't 'understand' parameters, added by new versions. In this connection, at installing new version and restoring more old versions you should take into account this specificity. So, before installing a new version you need to save configurations files of the current installed version.

Beginning from the version series 2.8.13 and later you can do saving and restoring configuration files using ViPNet program, working with configurations (see section 9.6). To meet this goal you can use partial configuration. For earlier versions of the software (up to 2.8.13) to solve described above problem before installing a new version you need to save manually the following configurations files in a temp folder:

iplir.conf, iplir.conf-<interface name>, mftp.conf, failover.ini

In versions up to 2.8.16 you can do saving and restoring configuration files using ViPNet program, working with configurations (see section 9.6). To meet this goal you can use partial configuration.

If after installing a new version, you have problems, described above, and you have decided to install the previous version over the new version you should restore manually saved configuration files (these files were saved manually or automatically) after the installing the old version. After that you can start ViPNet software services.

Beginning from version series 2.8.16 ViPNet software allows to do rollback to previous versions with automatic recovery of configuration. To do this you can use so-called rollback configurations, saved by setup program at installing ViPNet software. If there are no any previous versions the setup will save a version of software, you are installing. At installing over installed ViPNet software the setup program will save installed earlier software version to partial configuration (see section 9.6) with version number of installed earlier software, which this configuration is related to. This configuration will be named rollback-<year>-<month>-<day>. At next installations of old version over new one, the setup program will offer to restore a configuration, related to a version you are installing (if such a configuration was found) (see section 2.3).

In such a way, this mechanism makes easy manual storing and recovering of configurations at rollback to old versions of ViPNet software. For example, version 2.8.16 was installed, then 2.8.18 was installed over it, and then they decided to install version 2.8.17, that is more stable than version 2.8.18, but corrects critical errors of the 2.8.16. In this case, at installing version 2.8.17, the setup program will find saved rollback configuration of version 2.8.16 and will offer to restore it. This scenario will work correctly, because configuration files have upward compatibility, i.e. all parameters, used in older versions (even if these parameters are used no more) won't be errors. If there are several fit rollback configurations the setup program will choose a configuration with latest date of saving.

To work with configurations more effective it's not recommended to remove rollback configurations manually at rollback to old versions of the ViPNet software.

6 Configuring ViPNet Linux

6.1 General Configuration Principles

ViPNet Linux is configured by editing the configuration files iplir.conf and iplir.conf-<interface>. Before editing the files, you should stop the control program using the command iplir stop, then make the necessary changes and restart the control program using the command iplir start. It is necessary to stop the control program because it can update information in the configuration files whenever it becomes necessary in the process of its work. For example, once it receives information about changes in clients' IP addresses from other servers, it saves the updated information into iplir.conf. That is why the control program must not be running when you are editing the file.

The files iplir.conf, iplir.conf-<interface> and firewall.conf consist of several sections. Each of them contains several parameters. Each line contains either the name of a section or the name of a parameter together with its value. The name of each section is enclosed by square brackets and it is not allowed to use spaces, tab characters, etc. in these names. The line with the name of a section is considered the beginning of this section. A section may end either where another section begins or at the end of the file. The names of all sections, parameters, protocols, etc. (except the names of protected hosts) must be in lower case.

Any line in the file starting with the "#" character is considered a user comment and is not processed during the interpretation of the file. When the control program updates the file, comments are automatically placed at the beginning of the section they refer to.

Any line in the file starting with the ";" character is considered a service comment. These lines are sometimes added automatically when the control program is started or during its work and are used to inform the user about non-critical configuration errors, about filters enabled for some services (see below), etc. The user should not add service comments. They will be lost after the control program is started next time.

Each section contains one or several parameters. The name of a parameter is the first word in its line, then comes the "=" character, then a space and after that the value of the parameter. If the value consists of several parts, they are separated by a comma followed by a space. For the description of the firewall.conf configuration file syntax see section 6.3

6.2 Private network Settings

The private network settings are stored in the file iplir.conf.

The file iplir.conf may contain the following sections.

- [id] sections are used to describe address settings and access filters for some protected hosts. One [id] section corresponds to one protected host the local node can connect to. Also, one of the [id] sections contains the settings of the local node. Any [id] section contains the following parameters:

  * id – the unique identifier of the protected host. The control program uses this parameter to distinguish between [id] sections. The identifier of a protected host is the same for the entire corporate network, and you cannot modify this parameter. Each [id] section can have only one id parameter.

  * name – the name of the protected host. This parameter is specified by the ViPNet network administrator. It does not have any special meaning and is used

only to make configuration more convenient. But it is not recommended to change this parameter because it may cause problems (for example, for remote users viewing the traffic log of your computer: this log will have a modified name in it and it can confuse other users). Each [id] section can have only one name parameter.

* group – the name of the group the protected host belongs to. This parameter is specified if it is necessary to connect your protected host to a specified one via an alternative channel (see section 6.6 for the details). All the nodes with the parameter group and the same group name belong to one group. The nodes without this parameter don't belong to any group. The name of the group can be a random set of Latin alphabet symbols, figures and "minus", "underscore" and "dot" symbols. A group name is case sensitive. Group is an optional parameter and can be specified only once in the [id] section; if there are several parameters specified, it is considered an error. Moreover, this parameter must not be used in the [id] section for the local node, as well as in [id] sections for service filters. There is no group parameter in the [id] section by default.

* ip – defines the real IP address of the protected host. If there is a parameter secondaryvirtual (see below) with value on in the section [id] you should specify a corresponding virtual address through comma. The first parameter must be the real address and the next – virtual. For example, ip = 192.168.201.10, 10.1.0.5. See more about virtual addresses in the section 6.2.1. Each [id] section may have several ip parameters in case the described host has several network interfaces or several IP addresses on the interface. The closest address, on which there is an access to the corresponding host, must be specified first. When addresses are updated automatically, the closest address will be put in the first position automatically. When changing addresses sequence order, the conformance between real and virtual addresses will be kept (if secondaryvirtual=on), i.e. a virtual address will be rearranged like its real address. If only a real address is specified (no comma), that will mean a virtual address has not yet specified for this real address (in case of secondaryvirtual=on). Virtual address should not be specified manually, it is specified automatically.

* accessip – defines the current IP address for accessing the protected host from the local node, i.e. the IP address that the local node currently uses to connect to this host. It can take the value of either one of the real IP addresses of this host or the virtual IP address (see below) depending on the physical network topology and the working modes (see section 1.6) of your workstation and this host. This parameter is for informational purposes only and is used for the user to be able to find out during his work which IP address should be used to contact the remote protected host at the moment. It is determined automatically and it should not be edited manually.

* firewallip – in all [id] sections except the one of the local node this parameter defines the external IP address for accessing the host from the local node in case the host you connect to is behind a firewall, i.e. it works in one of the modes described in section 1.6. When you work with a host that is behind a firewall, all packets sent to it are encapsulated into a single type – UDP with the destination address equal to that specified in firewallip and with the destination port equal to that specified in port (see below). The type of the firewall that is used is determined by a number of additional parameters (proxyid, dynamic_timeout, etc.) in this section (see below). The usage of this parameter in the [id] section of the local node

is described in section 6.5 Each [id] section has only one firewallip parameter. If it is set to 0.0.0.0, it is considered that this protected host is not behind a firewall.

* port – in all [id] sections except the one of the local node this parameter defines the destination port to which packets should be sent for this host if it works in one of the firewall modes (see section 1.6). Each [id] section has only one port parameter. How to use this parameter in the [id] section of the local node is described in section 6.5.

* channelfirewallip – defines the external IP-address for accessing the alternative channel connecting your node with the specified node (see 6.6 for the details). Parameter setting format consists of two parts divided by a comma: a channel name and an external IP-address for the channel. The channel name must correspond to one of the channels specified in the [channels] section (see below). Setting any other parameter values is considered an error. This parameter can be set only in the [id] section of a coordinator. Moreover, setting several channelfirewallip parameters with the same channel name is also considered an error, as well as using this parameter in the [id] section for the local node and in [id] sections for service filters. This parameter is optional and is not used in [id] sections by default.

* channelport – defines the port parameters for accessing the alternative channel connecting your node with the specified node (see 6.6 for the details). Parameter setting format consists of two parts divided by a comma: a channel name and an access port for the channel. The channel name must correspond to one of the channels specified in the [channels] section (see below). Setting any other parameter values is considered as an error. This parameter can be set only if the channelfirewallip parameter is defined in the same [id] section for this channel. This parameter can be set only in the [id] section of a coordinator. Moreover, setting several channelport parameters with the same channel name is also considered as an error, as well as using this parameter in the [id] section for the local node and in [id] sections for service filters. This parameter is optional and is not used in [id] sections by default.

* proxyid – defines the firewall mode for its host (see section 1.6). In all [id] sections except the one of the local node, this parameter can take various values depending on the mode set on this host. How to use this parameter in the [id] section of the local node is described in section 6.55. For better perception, identifiers are written in the hexadecimal format both in the id parameter and in the proxyid parameter with the "0x" prefix before them. Each [id] section has only one proxyid parameter.

In most cases the values of the firewallip, port and proxyid parameters are specified automatically according to information received from other hosts. But in some cases, for example, when a VPN host works without an IP Addresses Server, they should be specified manually. If the value of any of the firewallip, port and proxyid parameters is zero for some host (except the local node), such a parameter is not specified in the configuration file.

* dynamic_timeout – defines the interval (in seconds) for polling the selected ViPNet Coordinator for incoming traffic to be passed through the firewall (see section 6.5.4). This parameter is present in all [id] sections except the one of the local node. This parameter is determined automatically and it should not be edited manually.

* usefirewall – in all [id] sections except the one of the local node this parameter defines the firewall settings for its host. It can take the values on and off. If it is set to off, the firewallip, port and proxyid parameters in this section are ignored and it is possible to work with this host only using one of its real IP addresses. For local node this parameter defines using external Firewall, i.e. if the value of this parameter is off, the external Firewall won't be use. On value must be used for the rest modes of thw work. How to use this parameter for modes settings of the local node is described in section 6.55.

* fixfirewall – present only in the [id] section of the local node. Defines the mode of applying firewall settings to the local node. It can take the values on and off. How to use this parameter is described in section 6.5.

* virtualip – the base virtual address of protected host. See more about this parameter in the section 6.2.1. Each [id] section has only one virtualip parameter. It should not be modified because virtual addresses are assigned automatically anyway.

* forcereal – allows you to use the real address of this host to connect to it, even when it is supposed to be visible only by its virtual address (see below about the rules of connecting to protected hosts). This parameter can take the value on or off, the value off is the same as if this parameter were not in the section (in this case this parameter is removed from the section when the control program is started next time). You should use this parameter with caution because the real addresses of hosts that are visible by their virtual addresses may coincide (if these hosts are located in private networks). If you specify the forcereal parameter for two hosts whose real addresses coincide, it may lead to unpredictable results.

* always_use_server – an attribute of operation of a node in the ***"Dynamic NAT"*** mode with traffic redirect through selected coordinator (see section 6.5.4). this parameter is intended for service and it should not be changed manually. The parameter takes on a value. "on" and presents only in the case of working this node in the specified mode.

* secondaryvirtual – defines assigning virtual addresses for this workstation. This parameter can take on the following values: on and off. The value off means this parameter is absent in the section (this parameter will be absent in the section at the next startup of the control daemon as well). If the value is on then a corresponding virtual address will be assigned to each real address, specified in the parameter id (see section 6.2.1). if this parameter is absent or its value is off, virtual addresses won't be assigned and work with the workstation will be realized with using base virtual address, defining by the parameter virtualip. In this case, all virtual addresses, presented in parameters ip are deleted from the configuration file. By default this parameter is absent.

* filterdefault – defines what to do with packets sent to this host or coming from it by default if they do not fall under more specific filters. The value of this parameter can be either pass (pass packets) or drop (block packets). Each [id] section has only one filterdefault parameter.

* blockforward – switches on/off blocking transit packets going from/to this node. This parameter can be on or off, off means it's absence in this section. By default, this parameter is absent for all nodes. Switching on this parameter means that all transit packets for this node will be blocked with code 70 (see section. 9.1).

it's allowed to use this parameter in sections of other nodes only. Using this this parameter in the section of own node and in the main and broadcast filters of protected network is an error; at this daemon won't be launched.

* filtertcp – defines rules for passing or blocking TCP packets. The value of this parameter consists of four or five parts separated by commas or spaces as described above. The first part contains the number of the local port of the TCP connection, while the second part is the number of the remote port of the TCP connection. In both cases you can specify a range of ports instead of one port, in this case the beginning and the end of the range are separated with the «minus» symbol. Note that, unlike in most widely used firewalls, these port numbers do not depend on the direction of the packet in ViPNet. They do not describe the port numbers of the sender and the recipient, but always describe the local and remote port no matter in what direction the packet is sent. The third part describes what to do with packets and it can take the value pass (pass them) or drop (block them). The fourth part describes the direction of the TCP connection and can take the values send, recv and any. And again note that this part describes not the direction in which a packet is sent, but the direction of the TCP connection, i.e. who was the client and who was the server when the TCP connection was being established. If it takes the value send, the rule applies to those TCP packets with the corresponding local and remote port numbers that belong to connections initiated by the local node, while the direction of the packets themselves does not matter. If it takes the value recv, the rule applies to packets belonging to connections initiated by the remote node. And if the value is any, the rule applies to any packets with the corresponding port numbers. The fifth part is optional and can take the value disable which indicates that this filter is temporarily disabled and the system works as if there were no such filter.

For instance, the rule

filtertcp= 22, 1024-65535, pass, recv

indicates that TCP packets belonging to connections established by the remote computer using a port number from 1024 to 65535 with port 22 on the local computer must be passed. At the same time these packets can be sent in both directions. On the other hand, this rule blocks packets when the local computer establishes a connection to port 22 on the remote computer.

* filterudp – defines rules for passing or blocking UDP packets. The value of this parameter has the same syntax as the value of the filtertcp parameter. But since the UDP protocol does not imply establishing a connection, the fourth part of the rule (that describes the direction) actually describes the direction in which each packet is sent, i.e. send applies to a packet sent from the local computer, recv applies to a packet sent to the local computer from the remote one and any applies to packets send in both directions.

filtericmp – defines rules for passing or blocking ICMP packets. The syntax of the value for this parameter is similar to that for the filterudp parameter, but the type and the subtype of ICMP messages are specified instead of port numbers. The first part contains the type, while the second part contains the subtype. You can also specify ranges of types and subtypes. The rest of the parts in the rule mean the same as in the filterudp parameter.

Each [id] section may contain as many filtertcp, filterudp and filtericmp parameters as necessary. When a packet comes from this protected host or is sent to it, the

filters are checked in the order they are specified. If the packet falls under some rule, it is processed according to it and filters are not checked further. If the packet does not fall under any rule, the operation specified in the filterdefault parameter is carried out.

* filterip – defines the rules for processing packets of IP protocols different from TCP, UDP and ICMP (for example, IGMP and others). The value of this parameter is the number of the IP protocol in question. Packets of this protocol are processed opposite to what is specified in filterdefault. If filterdefault is set to pass, packets of the protocol specified in filterip are blocked and vice versa. Each [id] section may contain as many filterip parameters as necessary.

* filterservice – defines the rule for processing packets belonging to some service. A service is a named set of rules filtertcp, filterudp, filtericmp (see below). The value of this parameter consists of two or three parts separated by commas. The first part is the name of the service each packet is checked to belong to. The second part describes what to do with packets – pass or drop. The third part is optional. It can take only the value inform. If it is specified, then, once started, the control program inserts service comments describing which particular set of rules this service represents after the filterservice line.

* tunnel – defines unprotected workstations tunneled by this server. It makes sense only for a protected host that is a server. The value of this parameter looks as follows: <ip1>-<ip2> to <ip3>-<ip4>, where ip1 and ip2 are the first and the last addresses of the tunneled range, ip3 and ip4 are the first and the last addresses used for mapping this range on the local network node. In most cases you should specify the same ranges, i.e. ip3 is equal to ip1 and ip4 is equal to ip2. But there are cases when the range ip1-ip2 belongs to a private network and the same private network addresses already exist in the local area network of the computer that is being configured. In this case the range ip1-ip2 is mapped to other, free addresses by specifying different ip3 and ip4 values. Actually, the ip4 value is ignored and generated automatically by adding the difference between ip2 and ip1 to ip3. For example:

tunnel= 192.168.201.5-192.168.201.10 to 192.168.201.5-192.168.201.10

means that this server tunnels addresses from 192.168.201.5 to 192.168.201.10 that are mapped on the local computer unchanged;

tunnel= 192.168.201.5-192.168.201.10 to 192.168.202.5-192.168.202.10

means that this server tunnels addresses from 192.168.201.5 to 192.168.201.10 that are mapped to addresses from 192.168.202.5 to 192.168.202.10;

The tunnel parameters are not sent over the network. It means that if some ViPNet server tunnels a set of unprotected workstations, other ViPNet servers and clients do not receive this information automatically. You should manually specify on each host that will use ViPNet to work with these tunneled workstation that this server will tunnel these workstations. An example of configuring tunnels using one of the typical schemes is given in **Appendix E. An example of configuring tunnels using the ViPNet Coordinator.**

Some [id] sections have a special meaning. The very first [id] section describes the local node. You can change its parameters by editing the parameters of this section. The changed parameters will be sent over the network then.

Some comments about configuring the [id] section of the local node:

- ✓ You should not change the ip parameters. They are automatically filled with values received from the operating system once the control program is started.
- ✓ By changing the usefirewall, firewallip, port, proxyid, fixfirewall parameters together with the parameters of the [dynamic] section (see below), you can set various firewall modes for the local node (see section 1.6). You can find how to configure various modes for the local node in section 6.5.
- ✓ If the local network node is a server and it tunnels some unprotected workstations, you should always specify the same values for the real range of addresses and for the range used for mapping in the tunnel parameter. It means that the tunnel parameter must be tunnel= ip1-ip2 to ip1-ip2. If you ignore this rule, the range for mapping addresses is automatically made equal to the real range of addresses.

You can also use the parameters in the [id] section of the local node to set various modes described in section 1.6 for the local node. You can find a more detailed description of how to set and configure these modes below. While configuring it, you should pay attention to routing settings. You should observe several rules:

- ✓ If the local node tunnels some workstations, you should set default gateway on these workstations to the address of this node.
- ✓ If the local node works with at least one other workstation using its virtual address, it should have the default gateway parameter specified. If no default gateway is specified, the system may block a packet on an early stage and it may fail even to reach the driver.
- ✓ If the local node is used as a proxy server, it must have the IP forwarding feature enabled no matter how many network interfaces it has.

Two [id] sections with special meanings follow the first one. Their id parameters differ from others and they are equal to 0xffffffff and 0xfffffffe. These sections are used for filters only. It makes sense to specify only the following parameters for these sections: filtertcp, filterudp, filtericmp, filterip, filterdefault. The section with id= 0xffffffff is responsible for broadcast packets sent by protected hosts. By specifying the corresponding filters in this section, you can forbid or allow certain types of broadcast packets to pass. The section with id= 0xfffffffe is the main filter of the private network. Rules specified in it are checked before filters for each particular protected host start being checked. If a packet falls under some rule, it is either passed or blocked at once and the rest of filters are not analyzed. In case a packet falls under filterdefault of the main filter and it is set to pass, filters for a particular protected host are analyzed. Setting filterdefault of the main filter to drop will result in all encrypted packets being blocked. A certain set of filters necessary for the normal work of a private network is automatically specified in the main filter and cannot be deleted.

Either real or virtual addresses can be used to connect to protected hosts described in [id] sections. No matter in what mode the local node works, it uses real addresses to connect to hosts it receives broadcast packets from. In other cases the type of address that is used to connect to a remote host is determined in the following way:

- ✓ If a local node is a ViPNet workstation, i.e. it does not act as a coordinator and works in the ***"No external firewall"*** mode (see section 6.5.1), i.e. it is not behind an external firewall, then:
  - ∗ Virtual addresses are always used to connect to the coordinator.
  - ∗ Real addresses are used to connect to ViPNet workstations working in the ***"No external firewall"*** mode, i.e. that are not behind any external firewall.
  - ∗ Virtual addresses are used to connect to ViPNet workstations working in any mode other than the ***"No external firewall"*** mode, i.e. that are behind some external firewall.
- ✓ If the local node is a ViPNet coordinator and works in the ***"No external firewall"*** mode (see section 6.5.1), i.e. it is not behind an external firewall, then:
  - ∗ Real addresses are used to connect to ViPNet workstations working in the ***"No external firewall"*** mode, i.e. that are not behind any external firewall.
  - ∗ Real addresses are used to connect to hosts working in the ***"ViPNet-coordinator"*** mode and using the local node as a ViPNet proxy server.
  - ∗ Virtual addresses are used to connect to all other workstations.
- ✓ If the local node works in the ***"ViPNet coordinator"*** mode (see section 6.5.2), i.e. it is behind some ViPNet proxy server, then:
  - ∗ Real addresses are used to connect to ViPNet workstations working in the ***"No external firewall"*** mode, i.e. that are not behind any external firewall.
  - ∗ Virtual addresses are always used to connect to coordinators other than the ViPNet proxy being used.
  - ∗ Real addresses are used to connect to hosts that are behind the same interface of the same ViPNet proxy server the local node is behind (their firewallip coincides with firewallip of the local node). The real address is also used to connect to this ViPNet proxy server.
  - ∗ Virtual addresses are used to connect to hosts that are behind one ViPNet proxy server with the local node, but behind a different interface. Virtual addresses are also used to connect to hosts working in any mode other than the ***"No external firewall"*** mode, i.e. that are behind some external firewall.
- ✓ If the local node works in the ***"Static NAT"*** or ***"Dynamic NAT"*** mode (see sections 6.5.3 and 6.5.4), i.e. it is behind some external firewall, then:
  - ∗ Real addresses are used to connect to ViPNet workstations working in the ***"No external firewall"*** mode, i.e. that are not behind any external firewall.

* Real addresses are used to connect to hosts that are behind one firewall with the local node (their firewallip coincides with firewallip of the local node).

* Virtual addresses are used to connect to all other workstations.

In all those cases described above when the virtual address must be used in order to access a protected host, its real address is used to access it if the forcereal parameter (see above) is specified for it.

As a rule, there is no need to memorize all host visibility rules described above. The current access address of a protected host is always displayed in the accessip parameter of the corresponding [id] section and this address must be used for any connections to this host.

- The [adapter] section describes network adapters installed on the computer. Each adapter must have its own [adapter] section. All packets on adapters not described in some [adapter] section are blocked. If no [adapter] sections are specified, the control program gets the list of adapters from the system and automatically creates [adapter] sections once it is started. When the control program is started after that, it works only with this list of adapters and does not detect a new adapter if it appears in the system. If new adapter appears in the system, you should manually add an additional [adapter] section to the control program configurator in order to enable ViPNet for it. It is enough to specify only the "name" and "type" parameters for this section (see below). The "ip" parameter is optional because its value will be obtained from the system next time you start the control program. In the process of its operation the control program periodically updates the parameters of the specified adapters at the interval defined by the "ifcheck_timeout" parameter of the [misc] section (see below). If it detects that some adapter is disabled in the system, it also disables it in the ViPNet protection system driver. If an adapter is enabled or if its address changes from its addresses' set, the control program loads this changes to the driver. Information about all events described above is written to the system log.

Each [adapter] section contains the following parameters (each of them must be present and only once):

* name – defines the system name of the adapter, for example, eth0, ppp0, etc. If several IP addresses on one network adapter are specified and one or some pseudo-devices (eth0:1, eth0:2 etc.) that regardless of this controlling daemon and driver all these devices will be the only one physical device with the base name (eth0).

* ip – defines the IP address of the adapter. This parameter is specified automatically and is used only for informational purposes. For each adapter several id parameters can be specified (for example, when using **aliases**).

* type – the type of the adapter for ViPNet. This parameter is present only if the host is a ViPNet coordinator, this parameter is not used for clients. This parameter can be set to either internal or external. This parameter is specified according to the following:

  ✓ If the host is not behind an external firewall or a ViPNet proxy, i.e. it works in the ***"No external firewall"*** mode (see section 6.5.1), the types of all adapters must be set to internal.

✓ If this host is behind an external firewall or a ViPNet proxy, i.e. it works in any mode different from the ***"No external firewall"*** mode, the type of one adapter which it will use to connect to the host working as a firewall is set to external, the types of other adapters are set to internal.

Each adapter described by the [adapter] section has its additional configuration file named iplir.conf-<name>, where <name> is the system name of the adapter specified in the name parameter of its [adapter] section. Below you can find mode details about configuring this file.

- The [dynamic] section contains parameters necessary for configuring the ***"Dynamic NAT"*** mode (see section 6.5.4):

  * dynamic_proxy – enables the ***"Dynamic NAT"*** mode for the local node. This parameter can take the value on or off, which corresponds to the enabling and disabling this mode. By default, the parameter is set to off. In the previous versions of ViPNet Linux this parameter was located in the [misc] section, and it is automatically moved to this section when you update the program to this version. The corresponding warning is written to the system log. You can find out how to select various modes for the local node in section 6.5.

  * firewallip – the external IP address for other hosts to access the local node working in the ***"Dynamic NAT"*** mode. In the previous versions of ViPNet Linux this parameter was located in the [id] section and named [dynamic_firewallip], it is automatically moved to this section when you update the program to this version. The corresponding warning is written to the system log. This parameter is determined automatically and it should not be edited manually.

  * port – defines the destination port where packets for the local node must be directed if it works in the ***"Dynamic NAT"*** mode. In the previous versions of ViPNet Linux this parameter was located in the [id] section and named dynamic_port, it is automatically moved to this section when you update the program to this version. The corresponding warning is written to the system log. This parameter is determined automatically and it should not be edited manually.

  * forward_id – the identifier of the ViPNet coordinator located in the external network and used to organize incoming connections to the local node working in the ***"Dynamic NAT"*** mode. The identifier is written in the hexadecimal format with the prefix "0x" before the value. The description of this parameter is available in section 6.5.4.

  * always_use_server – enables the mode when any traffic to external hosts is sent via the ViPNet coordinator specified in the forward_id parameter of this section, i.e. all connections to other hosts will be established ONLY via the external ViPNet coordinator. This parameter can take the value on or off. By default, the parameter is set to off. The description of this parameter is available in section 6.5.4.

  * timeout – the interval (in seconds) for polling the ViPNet coordinator in order to allow incoming traffic be passed through the firewall. By default, the parameter is set to 25 seconds. The description of this parameter is available in section 6.5.4.

- The [misc] section contains various additional parameters:

* packettype – defines the format of encrypted packets. Starting from version 3.4.0 only the version 4.1 packet format is supported. If you are updating from the previous version, where the version 4.0 packets were used, the format will be automatically changed to 4.1. The value of the packettype parameter affects only the type of packets sent by this host. The packet type is determined automatically for incoming packets and its decryption will be performed independent from the defined packettype value.

* ciphertype – defines the encryption algorithm for the outgoing packets addressed to ViPNet hosts. The parameter accepts the following values: gost – encryption with GOST algorithm; aes – encryption with AES algorithm. The ciphertype parameter applies only to the outgoing traffic encryption. The default parameter value depends on the following:

  * if key sets for the current host were created using the ViPNet Manager software which is a part of the ViPNet OFFICE software suite, the default parameter value is aes;

  * if key sets for the current host were created using the ViPNet Administrator [Network Control Center] software, the default parameter value is gost.

* timediff – defines the maximum time difference between protected hosts. Due to security reasons, ViPNet does not accept packets from a host if the difference between its time and the time at the local node is more than the number of seconds specified in the timediff parameter. By default, its value is set to 7200, i.e. two hours. If you work with hosts located in different time zones, this value should be increased.

* server_pollinterval, client_pollinterval – time intervals for polling inactive protected hosts. Protected hosts periodically exchange service packets in order to have information whether some host is working or not. Clients exchange this information only with their IP Addresses Server, while servers exchange it with each other. These parameters are present only in the configuration files of those hosts that are coordinators. The server_pollinterval parameter defines the time interval for this server to poll other servers. The client_pollinterval parameter defines the time interval for client workstations to poll their IP Addresses Server. These clients receive this parameter in each session. If no service packets have been received from a protected host that is supposed to exchange such packets with this host for the time interval specified in the corresponding pollinterval parameter, such a host is sent a special packet requiring a respond. If no respond is received, the host is considered switched off. The value of these parameters is specified in seconds and is set to 900 seconds (15 minutes) for servers (server_pollinterval) and 300 seconds (5 minutes) for clients (client_pollinterval) by default. Reducing the values of these parameters will make it possible to determine faster if some hosts are disabled, but it will increase the amount of service traffic and vice versa.

* iparponly – allows you to block packets that do not belong to an IP-protocol. ViPNet is used to analyze IP packets and passes all packets of all other protocols by default, which corresponds to the iparponly parameters being set to off. If you set this parameter to on, ViPNet will block packets of all protocols except IP, ARP and RARP. Packets of the ARP and RARP protocols are always passed because it is necessary for the IP protocol to function successfully.

* ifcheck_timeout – defines the time interval (in seconds) for polling the parameters of the system adapters known to the control program (see above). By default, the parameter is set to 30 seconds.

* warnoldautosave – defines whether to warn about old autosaved configurations of ViPNet (see section 9.6). It can take the values on and off. If the parameter is set to on, once the control daemon is started, warnings about automatically saved configurations will be displayed if the date of saving these configurations is more than one month earlier than current date.

* ipforwarding – managers switching 'IP forwarding' in the system. It can take on the following values: on – to forcedly switch on 'IP forwarding' when starting controlling daemon; system – not to change forwarding settings when starting. It's recommended to set on value, because when the forwarding is switched off, proxing and tunneling won't be able to work. It's recommended to use off and system values for debugging only. This parameter is present in the configurations file for coordinator only.

- The [servers] section contains the list of servers known to this host together with the information about which server is used as an IP Addresses Server for this host. This section contains the following parameters:

  * server – describes one of the known servers. The value of this parameter is a string containing the identifier and the name of this server separated by a comma. It is not recommended to change these parameters.

  * active – defines which server will be used as an IP Addresses Server. This parameter is specified only for ViPNet workstations, it is not specified for ViPNet coordinators. If the local node works in the ***"ViPNet coordinator"*** mode, i.e. it is behind an external ViPNet proxy server, you should specify this ViPNet proxy server as its IP Addresses Server. In other cases you can specify any server.

- The [channels] section defines a list of alternative channels for the defined node's access to ViPNet coordinators and the registration of groups of coordinators to work via alternative channels (see 6.6 for the details). The [channels] section is optional. If it is not defined, no channel is defined. If there is a [channels] section, but no channel parameter is defined in it, the section is deleted automatically. This section can include one or several channel parameters. The format of a channel parameter should be the following: **<channel name>, <a list of working via this channel node groups, divided by commas>.** A node name is the node's unique identifier. Node group names are defined in [id] sections (see above). Here is an example of setting parameters in this section:

```
[channels]
channel= LocalNet, WorkDepartmentGroup, OtherGroup
channel= ReservLocalNet
channel= Internet, SalesDepartmentGroup
```

A channel name can be a random set of Latin alphabet symbols, figures and "minus", "underscore" and "dot" symbols. A channel name is case sensitive. A list of group names after a channel name is optional.

There can be the following configuration errors for the [channels] section:

- A group not set in any [id] section is defined for some channel parameter;
- The same group name is defined for two or more channel parameters, i.e. a node group can't be simultaneously registered for several channels;
- The same channel names are defined for two or more channel parameters.

You can find the detailed description of the procedure of working with a remote coordinator via a defined channel in section 6.6.

- The [service] section defines one of the services. A service is a named set of filters that are used for some protected hosts as a whole by specifying the filterservice parameter (see above). Any number of [service] sections can be present in the configuration file. It is recommended that these sections precede any other sections. At least, they must be specified before they are used in the filterservice parameters. After the control program is started, all service definitions are automatically moved to the beginning of the file. The [service] section can contain the following parameters:

  * name – defines the name of the service. Latin letters, digits, hyphens and underscores can be used in these names. This parameter must be present and the name of each service must be unique within each configuration file.

  * parent – defines the name of the service that is parent for this service. This parameter is optional. If it is specified, its value must be either the name of some standard service (see below) or the name of a service defined prior to this one in the same configuration file. The child service contains all filters defined for its parent and you can also define additional filters in it. You cannot delete any filters of the parent service in its child service. It is possible to specify several parents, in this case the child service inherits all their filters. A child service can be used as a parent one for other services. Such child services inherit the entire set of filters from all their parents. The level of nesting is not limited.

  * filtertcp, filterudp, filtericmp – define the set of filters this service will contain. These parameters have the same syntax as the corresponding parameters in the [id] section (see above). The difference is that these parameters in the [service] section have no third part describing what should be done with packets – pass or drop. This part remains empty and, once specified in filterservice, takes the value specified in filterservice.

Sample service description:

```
[service]
name= http
filtertcp= 1024-65535, 80, , send
filtertcp= 80, 1024-65535, , recv
```

Besides user-defined services, there is a set of the so-called standard services defined in ViPNet. The standard services are described in the file iplir.serv and cannot be modified by the user. The standard services are considered as initially defined in all configuration files and they can be used in the filterservice parameters and specified in the parent parameters of the [service] section to create user-defined services based on them.

- The [virtualip] section stores the settings of virtual addresses (see section 6.2.1). It contains the following parameters:

* startvirtualip – specifies the address to start generating base virtual addresses from. If the user modifies the startvirtualip parameter, all base virtual addresses are assigned anew, the same as when the configuration files were initially generated. Also you can assign virtual addresses in parameters ip for nodes using secondaryvirtual=on.

* startvirtualiphash is a service parameter. You shouldn't change this parameter unless absolutely necessary, except when fine-tuning for the purpose of reassigning virtual node addresses (see section **Fehler! Verweisquelle konnte nicht gefunden werden.**).

* endvirtualip – a service parameter. It contains the base virtual address that follows last assigned one. This parameter is used as a reference point, this address will be assigned to the next new node (workstation) if it will be added when searching and assigning base addresses for new nodes. You should not modify this parameter without any urgent necessity, especially to a larger value.

- The [debug] section defines daemon logs parameters (see section 13). It contains the following parameters:

  * debuglevel – debugging level, a number from -1 up to 5, by default 3. the parameter value -1 switches off the logging.

  * debuglogfile – is an identifier, defining log location. The format of this identifier is the following: **<protocol specifier>**, **<URL specifier for this protocol>**. See 13 for the detailed description of possible values of this parameter. The default parameter value is /var/log/iplircfg.debug.log, that corresponds to the logging to a specified file.

6.2.1 Assigning Virtual Addresses. General Overview

Virtual addresses are used for workstations, located behind external Firewall. Using virtual addresses is necessary because workstations, located behind different Firewalls, can have identical addresses in their private networks, and there will be an ambiguity when using real addresses. Virtual addresses are not used for workstations, not located behind external Firewall (except coordinators, working in the ***"No external firewall"*** mode, see section 1.6), but all such workstations have their stationary virtual address.

The four octets, forming IP address, are used at assigning virtual addresses by the following way:

- The first octet of all virtual addresses has identical value, corresponding to the first octet of start virtual address startvirtualip, specified by a user.
- The third and the fourth octets define a network node. They are identical for all virtual addresses of given network node and they are different for virtual addresses, belonging to different network nodes.
- The second octet defines one of the real addresses of a network node.

In other words, at first, the fourth octet will be changed from node to node, and then the third octet will be changed. The second octet will be changed from one real addresses of a node to another real address of the same node. The first octet will stay permanent.

It should be noted, that an assigning virtual addresses for each real IP address will be done only if the parameter secondaryvirtual=on for this work station (см. п. 6.2). Otherwise, a work with this workstation will be with using the base virtual address only.

A virtual address of a network node, in which the third octet is equal to the third octet of start virtual address startvirtualip, is called base virtual address virtualip of a network node.

Base virtual address is a reference point at assigning virtual addresses for each of real addresses.

The rest virtual addresses, featuring each of real addresses of the network node, are called secondary virtual addresses or simply virtual addresses. These virtual addresses should be specified through comma after real address in the ip parameter. One of the virtual addresses can be identical to base virtual address; such a situation will always take place, if a network node has one real address.

As it was mentioned above, a current access address is defined automatically and included in the accessip parameter. If at the moment a node is seen by virtual address, that its access address will be either a base virtual address or in case of secondaryvirtual=on, secondary virtual address, corresponding to the first real address in the list.

Secondary virtual addresses can be used if necessary to address by specific real address of this network node that can be visible by virtual addresses, and not to address to the node in general. Such a necessity can be arisen, for example, if an application, working on this node, waits for network requests only from one address of this node. In such cases you need to specify parameter secondaryvirtual=on. In most cases it's enough to specify one virtual address (the base one) to address to this node, and parameter secondaryvirtual should be specified in that cases when the node administrator neatly understands a necessity of such a specifying.

At link files updates and changes in real addresses list for any node, workstations will keep their virtual addresses and new added workstations and real IP addresses will get new vacant virtual addresses.

6.3 Configuring open packet processing rules

The configuration files of interfaces have the following sections:

Open (non-encrypted) packet processing rules are contained in the firewall.conf file. These rules may be divided into several types:

* Anti-spoofing rules
* Packet filtering rules
* Address translation rules

Below we discuss each of the types in more detail.

The following terms are used in the further description to specify addresses and address ranges: address, address range, and address mask, and port and port range.

An address is a usual IP address in decimal notation with separated by periods, e.g.: 192.168.1.1

An address range is two addresses separated with the "dash" symbol, which is recorded in the configuration file using the "minus" symbol. Here the second address (the end of the range) should exceed the first address (the beginning of the range). A range includes all addresses between the range beginning and the end, and the beginning and the end themselves. E.g.: 192.168.1.1-192.168.1.10

An address mask is a network address in the CIDR (Classless Internet Domain Routing) format. A mask contains a subnet address recorded in the usual IP address format, and a number of MSBs in the subnet mask that are equal to 1. The subnet address and the number of bits are separated with a "forward slash" symbol. E.g.: 192.168.1.0/24 describes a network with the 192.168.1.0 address and the 255.255.255.0 subnet mask.

A port is a number from 1 to 65535, describing a port for the TCP or UDP port.

A port range is a starting and ending port separated with the "dash" symbol, which is recorded in the configuration file using the "minus" symbol. E.g.: 1024-65535.

6.3.1 Firewall settings

The [settings] section of the firewall.conf firewall configuration file contains firewall parameters, that are not filtering rules. When the firewall.conf file is created, the [settings] section doesn't contain any parameters, and default values specified below are used. The following parameters are possible:

- max-connections – the maximum number of connections. This parameter limits the maximum number of simultaneous connections based on the available memory size. E.g., if we have 128 Mb of memory, the firewall may support about 30,000 simultaneous connections.
- dynamic-ports – a range of ports that are used for dynamic address translation.
- listen-timeout – a timeout for "auxiliary" connections. Some protocols require additional connections during operation. If the corresponding application protocol processing module is installed, then auxiliary connection rules are created automatically. This rule is removed, if no connection has been established during the period specified in listen-timeout.
- connection-ttl-tcp – a time period in seconds that has to elapse after the registration of the last packet related to this TCP connection, for the the connection to be broken according to the timeout. By default, this parameter is equal to 3600 (60 minutes).

- connection-ttl-udp – a time period in seconds that has to elapse after the registration of the last packet related to this UDP connection, for the the connection to be broken according to the timeout. By default, this parameter is equal to 300 (5 minutes).
- connection-ttl-ip – a time period in seconds that has to elapse after the registration of the last packet related to this IP connection, for the the connection to be broken according to the timeout. By default, this parameter is equal to 60 (1 minutes).
- dynamic-timeouts – may be equal to yes or no, and enables/disables the dynamic connection timeouts mode. This mechanism is used to counteract flood-attacks, and works as follows: when the number of connections reaches a certain percentage of the maximum, the timeouts of all connections are decreased by a certain value, and this value is the higher, the closer the number of connections is to the maximum (but the timeout is not decreased below a certain minimum). When the number of connections falls to a certain percentage of the maximum, the timeouts are restored to their normal value. The dynamic timeouts mode is disabled by default.
- cleanup-interval – a frequency in seconds, with which the firewall kernel performs actions to cleanup obsolete connections (with expired timeouts). Large values lead to a less accurate (in time) cleanup of obsolete connections, and very small values lead to a higher processor load. By default, this parameter is equal to 5 seconds.

6.3.2 Anti-spoofing

Anti-spoofing rules allows you to specify IP address ranges, packets from which are allowed on a particular interface. In this case packets from outside the allowed range will be blocked. Additionally, if some interface receives packets from addresses specified as allowed for another interface, the packets will also be blocked. As seen from the name, the main purpose of anti-spoofing is the protection against the so-called "spoofing", which is one of the network attack types. When spoofing, an intruder sends a packet to a computer, and the packet contains a sender address, which is not the address of the intruder, but some other address known to the computer. E.g., this makes possible to send a packet from the Internet to a gateway, by specifying as a sender address an address from a private LAN, which is also connected to the gateway. In this case the intruder may gain access to some service, access to which is only allowed from within the intranet. Anti-spoofing rules allow to exclude such possibility.

From the anti-spoofing operation's point of view, all network interfaces are divided into two types: internal and external. External interfaces include the interfaces connected to the Internet, and internal interfaces include the interfaces connected to the LAN (which usually has private addresses).

Anti-spoofing parameters are configured in the [antispoof] section of the firewall.conf file. First of all, this section contains the antispoof parameter, which may be set to yes or no to enable or disable the anti-spoofing mechanism, respectively. The other parameters have names corresponding to network interfaces' names. The values of these parameters have the following structure: first, an interface type is specified (internal or external), then a list of addresses, address ranges, or address masks allowed on this interface, separated from the interface type by a comma. The allowed address list elements, if there are several of them, are also separated by commas.

The list of allowed addresses for external interfaces may also include the anypublic keyword, which denotes "all interfaces allowed in the Internet". This term includes all addresses except those dedicated for special purposes: for the local network interface (127.0.0.0/8) and for private networks (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16).

The anti-spoofing section should always contain a list of all network interfaces except the local one (loopback). The local interface is not affected by any packet processing rules, and any packets are always allowed on it. Additionally, we should note that packets containing sender

addresses from the 127.0.0.0/8 range are blocked on all interfaces processed by the ViPNet Coordinator Linux, regardless of anti-spoofing settings, since this is a requirement of standards.

An example anti-spoofing section:

```
[antispoof]
antispoof = yes
eth0 = external,anypublic
eth1 = internal,192.168.1.0/24
eth2 = internal,192.168.201.1-192.168.201.255
```

In this example, anti-spoofing will operate as follows:

- eth0 – an external interface connected to the Internet. It allows packets from all addresses, except private ones.
- eth1 – an internal interface, which allows packets from addresses from 192.168.1.1 to 192.168.1.255.
- eth2 – an internal interface, which allows packets from addresses from 192.168.201.1 to 192.168.201.255.

If, in this example a packet from the Internet arrives to eth0 containing a sender address from the 192.168.1.0/24 or 192.168.201.0/24 network, it will be blocked. If a packet arrives to the eth1 interface containing a sender address from the 192.168.201.0/24 network, it will be blocked. Thus, reliable anti-spoofing protection is provided.

6.3.3 Packet filtering

After passing through the anti-spoofing mechanism, packets are processed according to packet filtering rules. These rules are divided into 3 rules tables, which are described by the corresponding sections. The first table contains rules for processing local packets, i.e., those containing the local computer as a sender or recipient. These rules are described in the [local] section. The second table contains rules for processing transit packets, which only pass through the computer on their way from the sender to the recipient. This table is described by the [forward] section. The third table contains rules for processing broadcast packets. These rules are described in the [broadcast] section.

Each section contains a set of rules, which have the same syntax for all sections.

If you need to allow transit packets through the ViPNet Coordinator Linux computer, you should either explicitly allow their passing in the [forward] table, or configure the corresponding interfaces in modes allowing them to pass packets: the interface to which transit packets arrive in mode 4, and the interface from which the packets depart in mode 3 or 4. If other modes are used, it is necessary to create a permissive rule in the [forward] section. This relates to the case when address translation is used, more detail on setting this configuration below.

Each separate rule is described by a rule parameter, the value of which is the rule itself. A rule contains several components:

- a control component
- a condition
- an action
- a schedule

Each of these components, in turn, may contain parts, that we'll call lexemes. A lexeme is a service work, which may be followed by a parameter. Rule components, the lexemes forming them, and parameters and service words within the lexemes are separated by spaces.

To process IP protocols different from TCP/UDP/ICMP virtual connections are created, based on IP addresses and the protocol number. Thus, it is enough to specify permissive rules only for the request initiating the connection, and responses will be received automatically (if they arrive from the IP address, to which the request was sent, and over the same protocol).

6.3.3.1 Control component

The control component describes rule properties, which are not directly related to packet processing, and is always specifies at the beginning of the rule. It may contain the following lexemes:

- num <number> specifies the number of the rule in the table (0 to 65535). The number are used to denote rules' priorities, the less the number, the higher the the priority. During packet processing, first the higher priority rules' conditions are checked, and in case of match, the action specified in the rule is performed, and the further rules browsing is stopped.

  The num lexeme may be omitted, in this case the ViPNet Coordinator Linux will try to assign a rule number on its own, based on the number of the rules located before and after the rule. This doesn't always lead to an expected result, so it is recommended to explicitly specify the number for each rule.
- disable specifies that the rule is temporary disabled and is not applied. If the num lexeme is specified, then the disable lexeme may be only specified after it; if the num lexeme is omitted, then it may be specified in the beginning of the rule.

The remaining rule components may be specified in any order, but you shouldn't alternate the lexemes that form them, i.e., specify a condition lexeme, an action lexeme, then another condition lexeme, etc.; the lexemes of each component should be specified continuously.

6.3.3.2 Condition

The condition specifies the parameters the packet should have to be processed by this rule. A condition may contain the following lexemes:

- proto <protocol> specifies the transport protocol the packet should belong to. Supported protocols include tcp, udp, and icmp, you may also specify digital numbers of any protocols. If you need to process packets from different protocol in a single rule, you may specify them separated by commas. Additionally, you may specify the any keyword instead of a protocol, which means all protocols. But it should be noted that when specifying any protocol, you can't specify port numbers in the condition, since not all protocol have the conception of a port. For the [broadcast] table you may specify udp and icmp only. Filtering rules for the icmp protocol also can't contain conditions for ports (the port lexeme).
- from <address list> specifies conditions for packet sender address and port. If both an address and a port are specified, they should be separated by a colon (e.g.: 192.168.201.1:22). If a port is not specifies, the colon is not used, in this case the condition is applied to all ports. In addition to a single address, you may specify an address range or an address mask (e.g.: 192.168.1.1-192.168.1.10:22 or 192.168.201.0/24:22). You may also specify a port range (e.g.: 192.168.201.0/24:1024-65535). Also, you may specify several "address + port" conditions separated by commas (e.g.: 192.168.1.1-192.168.1.10:22,172.16.1.0.24:25). Additionally, you may combine addresses, ranges, and masks and ports and port ranges into groups, listing them separated by commas, and including a group into parentheses.. In this way you may link several address ranges or masks to a single port range, without repeating it several times. E.g.: (192.168.201.0/24,172.16.1.0/24):1024-65535 means "all packets from all

addresses in the 192.168.201.0/24 and 172.16.1.0/24 networks having a sender port from 1024 to 65535". Even more complex notation forms are possible, with a simultaneous grouping of addresses and ports, and with a listing of such groups. E.g.,

(192.168.201.0/24,172.16.1.0/24):(22,25,6660-6667),10.0.0.0/8:1024-65535

will mean "all packets from all addresses in the 192.168.201.0/24 and 172.16.1.0/24 networks, having a 22, or 25, or 6660 to 6667 sender port, and packets from addresses in the 10.0.0.0/8 network having a 1024 to 65535 sender port.

You may also specify the following keywords instead of addresses and their ranges:

- any means all addresses (i.e., the 0.0.0.0-255.255.255.255 range)
- broadcast means the 255.255.255.255 address

- to <address list> specifies condition for packet recipient address and port. The syntax is the same as for the from lexeme.

  You may only specify the following addressed in the to field of the [broadcast] table:

  - broadcast the 255.255.255.255 address
  - broadcast addresses of subnets connected to the network interfaces of the computer. Giving a specific broadcast address affects directed broadcasts sent in the corresponding subnet.
  - directed-broadcast broadcast addresses of all subnets connected to the interfaces of the computer. When loading a rule into the driver, this keyword is replaced with a list of corresponding broadcast addresses.

- in or out specifies a connection establishment direction. When specifying these conditions you should remember, that they specify not the direction of a separate packet travel, but the *connection establishment direction*. The ViPNet Coordinator Linux traces, which packets belong to which established connections, and allows or rejects them according to rules. E.g., if we have the following condition

  proto tcp from 192.168.1.1 to any out

  then it will apply to all local packets related to connections established from the 192.168.1.1 address; if a packet arrives from a remote station to the 192.168.1.1 address, the rule also applies to it. Bu if a remote computer tries to establish a connection to 192.168.1.1, then packets related to this connection will not correspond to the above condition.

  Connections for the TCP protocol are always traced. For the UDP protocol, which doesn't have a connection concept, an attempt to trace a virtual connection, which is established between applications using UDP in most cases, is also made. E.g., if we have the following condition

  proto udp from 192.168.1.1 to any:53 out

  then, after the 192.168.1.1 computer sends a UDP packet to port 53 of a remote computer, it is considered, that a virtual connection is established to it. Then, if a reply from the remote computer arriver shortly to the same port that was used for sending the first packet, this reply also corresponds to the above condition, as belonging to the established virtual connection. Virtual connections are considered broken, if they have no traffic during a time period defined for the protocol (see Firewall settings).

  In cases when applications exchange UDP packets using different port numbers for sending and receiving packets, there is no way to trace a virtual connection. In such cases you should view the in and out lexemes ac corresponding to the packet travel direction.

The proto, from, and to lexemes should always be specified in a condition. If some of these parameters is not important, you should specify any. A direction may be omitted, in this case it is considered that the condition applies to connections established in both directions.

Example complete conditions:

proto any from any to 192.168.201.1:22 in

proto tcp,udp from any:53 to 192.168.0.0/16,172.16.1.0/24

proto tcp from 10.0.0.1 to (192.168.0.0/16,172.16.1.0/24):(22,25) out

6.3.3.3 Action

The action describes what needs to be done to a packet, whose parameters conform to the condition. An action consists of a single lexeme, which may have the pass pass value, if the packet should be allowed, or the drop pass value, if the packet should be dropped (blocked).

6.3.3.4 Schedule

The schedule allows you to set time periods when the rule is applied. When a schedule is omitted, the rule is applied permanently. A schedule is described by a single time lexeme, whose parameter contains several comma-separated parts.

The first part is called the schedule mode, and may have one of the following values: on, off, disable. The on schedule mode means that the rule is applied during time periods specified in the schedule, and not applied during other time periods. The off mode, otherwise, means that the rule is not applied during the specified time periods, and is not applied during other time periods. If, however, the schedule mode is disable, then the schedule is disable, and the rule is always applied is if there were not schedule.

The second part is called the schedule type, and may have daily or weekly values.

A daily type schedule is a schedule for a day. With this schedule type you may specify a single time interval, during which the rule will be enabled (or disabled, depending on the schedule mode; only the on schedule mode is described for the sake of brevity), and each day, the rule will be enabled at the beginning of the specified period and disabled at its end.

The weekly schedule type allows you to specify a schedule for a week, and specify a particular time period during which the rule will be enabled for each day. This scheme allows you to account, for example, for another operating mode during weekends.

The third part essentially specifies time intervals.

If the schedule has a daily type, then a single interval is specified as hh:mm-HH:MM, where hh:mm are interval beginning hours and minutes and HH:MM are those of its end. The interval beginning time is included into the schedule operating time, but the ending time is not. Minutes may have values 0 to 59, and hours 0 to 24 (if the number of hours equals 24, then the number of minutes may only be equal to 00, and this time means 0 hours of the following day).

If the schedule has a weekly type, then the third part is used to specify a schedule for each weekday. For each weekday, you should specify the first three letters of its English name (mon, tue, wed, thu, fri, sat, sun), then an equals sign and a time interval for this day in the same format as was used for a daily schedule (e.g., mon=9:00-18:00). Schedules for different weekdays are separated by commas (e.g.: mon=9:00-18:00,tue=10:00-18:00). You may also specify the same time for several weekdays, in this case these are listed before the equals sign, separated by colons: sat:sun=00:00-24:00). You may omit some weekdays from the schedule, in this case during the omitted days, the rule will behave in the same way as for specified days outside specified time intervals (i.e., be disabled if the schedule mode is on, and be enabled if the schedule mode is off).

Example complete schedules:

time off,daily,9:00-18:00

time on,weekly,mon:tue:wed:thu:fri=9:00-18:00,sat:sun=00:00-24:00

6.3.4 Address translation

The ViPNet Coordinator Linux supports address translation, i.e., changing the packet's sender or recipient address according to certain algorithms. Two types of address translation are supported:

- Forward translation, also called masquerading or dynamic translation. Such address translation is used, if you need to grant Internet access to users, who have private addresses. In this case, when packers from private users pass through the Coordinator, their sender address is replaces with the external ("real") address of the Coordinator. When reply packets arrive, their recipient address is replaced back with the private address, and the packet in this way is delivered to the private network.
- Backward translation, also called port forwarding or static translation, is used when you need to provide access to a computer located in a private network from the Internet. In this case all packets arriving from the Internet to a certain port of the Coordinator's external address are redirected to the specified intranet address by substituting their recipient addresses, and the reply packets from the intranet computer have their sender address replaced.

6.3.4.1 Address translation rules syntax

Address translation rules are described in the [nat] section, and have a syntax similar to that of filtering rules. They are also described by the rule parameter, whose value also consists of a control component, a condition, an action, and, optionally, a schedule.

The control component is completely identical to the one described for filtering rules.

As an action, the [nat] table contains the change lexeme with a parameter specifying what to replace and with what. Depending on the translation type we want to perform, the action may be as follows:

- For forward translation, an action looks as follows: change src=<address>:dynamic, where <address> is the external address of the Coordinator, with which a packet sender address will be replaced. E.g.: change src=194.87.0.8:dynamic
- For backward translation, an action looks as follows: change dst=<address>:<port>, where <address> and <port> are the address and the port of a LAN computer to which packets will be redirected. E.g.: change dst=192.168.201.1:8080

A condition for the [nat] has roughly the same syntax as for filtering rules, but with some additional characteristics:

- If the protocol list contains tcp or udp (including the case when any protocol is specified), then the src lexeme in the substitution should always contain the dynamic port. It is not allowed prohibited to specify another port or to omit it completely. If protocols do not include tcp and udp, then, on the contrary, it is prohibited to specify a port in the substitution of the src lexeme. Here, in the first case, the substitution is interpreted as follows: the dynamic port is used for the protocols where it matters (and not for all specified).
- For forward translation, the from lexeme specifies an intranet address set to be translated, and you may only specify addresses, or ranges, masks, and lists of those in the from lexeme. It is not allowed to specify ports or port ranges. Additionally, for forward translation, the to lexeme should have the any value, and the port in the change lexeme should have the dynamic value.
- For backward translation the from lexeme should have the any value, and the to lexeme specifies the external address and port of the Coordinator, to which packets subject to redirection will arrive. In this case you may specify only an address or an

address list, and a port or a port range in the to lexeme. Specifying address ranges and masks is not allowed.

A schedule for translation rules is completely identical to that of filtering rules.

Example complete rules for the [nat] table:

For forward translation:

rule = num 10 change src=194.87.0.8:dynamic proto tcp from 192.168.201.0/24 to any

For backward translation:

rule = num 100 change dst=10.0.0.7:8080 proto tcp from any to 194.87.0.8:80

6.3.4.2 Interaction between filtering rules and translation rules

Packets subjected to address translation are also processed by filtering rules specified in the [forward] and [local] tables. When the correspondence between the parameters of a packet that has passed through address translation, and the condition of a filtering rule, the following principle is applied: the sender address is taken from the packet before translation, and the recipient address is taken from the packet after translation. It is these addresses that are checked for the compliance with rules' conditions.

Lets consider and example of this. Assume there is a Coordinator with a 10.0.1.0/24 intranet and a 194.87.0.8 external address, and we need to provide intranet users with Internet access. For this, the [nat] table should look as follows:

[nat]

rule = num 10 change src=194.87.0.8:dynamic proto tcp from 10.0.1.0/24 to any

You should also specify a permissive rule in the [forward], that will look as follows:

[forward]

rule = num 100 pass proto tcp from 10.0.1.0/24 to any

If here, for example, you need to prohibit internal users from establishing TCP connections with the 194.226.82.50 external address, add the following rule to the [forward] table:

rule = num 90 drop proto tcp from 10.0.1.0/24 to 194.226.82.50

As seen, the from contains the address of the local packet sender before translation, and the to contains the address of the remote recipient after address translation (in this case it is not changed during translation).

The same principle is applied during backward translation. Assume the Coordinator has a 194.87.0.8 external address, and we need to redirect all packets arriving to port 80 of this address to the LAN, to address 10.0.1.1 and port 8080. The [nat] will look as follows:

[nat]

rule = num 10 change dst=10.0.1.1:8080 proto tcp from any to 194.87.0.8:80

The [forward] will look as follows:

[forward]

rule = num 100 pass proto tcp from any to 10.0.1.1:8080

Now, if we need to prohibit incoming connections to this intranet computer from the 194.226.82.50 external address, we need to enter the following rule to the [forward] table:

rule = num 90 drop proto tcp from 194.226.82.50 to 10.0.1.1:8080

6.3.4.3 FTP connection address translation characteristics

Connections established over the FTP protocol are processed by the ViPNet Coordinator Linux in a special way. This is due to the fact that this protocol creates additional connections to dynamically assigned ports during its operation, which always leads to problems with firewalls. The ViPNet Coordinator Linux contains a special application processing module, which allows it to analyze the controlling FTP connection and to trace dynamic port numbers used for additional

connections. Here, if controlling connection establishment is allowed, then additional connections created after will also be allowed automatically, without any additional settings. This takes place when the FTP operates in both active and passive modes.

Such FTP protocol processing is implemented only for connections using a receiver port with number 21, which is dedicated for FTP servers. If an FTP uses a non-standard port, then port numbers of additional connections will not be traced, and these connections will not be allowed without additional configuration.

Additionally, tracing dynamic port numbers only operates for transit packets. If the FTP server is installed on the same computer where the ViPNet Coordinator Linux is running, and filtering rules only allow connections to port 21, then clients will be only able to work with this server in the active mode, if the corresponding adapter is in mode 3, or won't be able to work at all if the adapter is in mode 2. To allow working in active FTP mode when the adapter in in mode 2, you should allow outgoing connections from local port 20. To provide the operation of the FTP server in the passive mode, you should allows incoming connections for all local ports above 1024. There are FTP servers (such as ProFTPd), which allow limiting the dynamic port range used by the server; in this case you may allow incoming connections for this port range only. But generally, installing additional services, including FTP, to a ViPNet Coordinator creates potential security problems, and we don't recommend using such a scheme.

6.4 Configuring network interface parameters

Configuration files for network interfaces contain the following sections:

- The [mode] section is used to specify the security level of the ViPNet driver for this interface.

  * mode – defines the number of the security level for the interface. A more detailed description of security levels was given above (see section 1.5). Let us briefly enumerate available levels:

    - ✓ level 1 – pass only encrypted packets.
    - ✓ level 2 – pass encrypted packets and unencrypted packets from the computers explicitly specified in the public network filters.
    - ✓ level 3 – boomerang. In addition to level 2, pass all outgoing packets, as well as incoming packets (within a certain time interval) from hosts connections to which was established by outgoing packets.
    - ✓ level 4 – pass all packets.
    - ✓ level 5 – disable the driver, do not decrypt encrypted packets.

- The [db] section is used to specify the parameters of the traffic log. Separate log files are created for each interface and they are stored in the same directory where the configuration files are stored. Log files have the following names: iplir.db-<interface>. The user specifies the maximum log size. Records about packets are saved to the corresponding log file until it the maximum log size is reached. Then new records overwrite the oldest ones. To reduce the size of log files and make viewing them more convenient, similar records about packets registered within a short period of time are combined into one record and later you can find out how many times the event described by such a record took place while viewing the log. The [db] section contains the following parameters:

* maxsize – the maximum size of a log file in megabytes (1 megabyte = 1048576 bytes). The actual size of log files is approximately 1 KB bigger due to the service header. After the control program is started, the word Mbytes is automatically added after the log size to make it easier to understand this parameter. When you edit the file later, you can either delete or leave this word, it will be added automatically anyway. If you set this parameter to 0, no packet log will be kept for this interface. If there had been any records in the log before you set the size to 0, they are not deleted, but you cannot view them. The default value for the parameter is 50 Mbytes

* timediff – the time interval in seconds within which similar events are combined into one event in the log file (aggregation). The time is specified in seconds. The default value is 60 seconds. In case of specifying zero value the aggregation will be switched off (events won't be combined). If the aggregation is switched off and if traffic is very intensive, not all packets will be logged.

* registerall – defines whether all packets passing through the system should be registered. This parameter can take the value on or off. When it is set to off, only blocked packets are registered together with the events occurring when the addresses of protected hosts are changed.

* registerbroadcast – defines whether broadcast packets should be registered. Its possible value is on or off.

* registertcpserverport – defines whether the client's port should be registered for TCP connections. As a rule, the client's port is assigned automatically for TCP connections and it does not contain any useful information. If some host tries to connect to some port on the local node and the connection is not established for some reasons, next time the same host tries to connect, it will use another port, etc. When port scanners are used or some network attacks are in progress, the number of such attempts may achieve several hundreds per second. Since the client uses different ports each time, such packets are not considered similar and a separate record is created in the log for each of them. Such records clutter up the log file and complicate its analysis. When the registertcpserverport parameter is set to on, the port of the client is neither registered nor taken into account for TCP connections, which allows the events of connecting to some port on your host from a certain address to be combined into one record, which is often quite convenient.

6.5 Configuring Firewall Modes

If a ViPNet host works within an internal network with internal IP addresses and there is a firewall or some other device translating internal addresses into addresses available in an external network (that is, Network Address Translation is done) set up on the border of this internal network, one of the firewall modes must be configured to make proper work with other hosts outside the internal network or behind other firewalls possible. The rules of selecting various modes are described in section 1.6. When you are selecting a mode, you should remember that the controlling daemon must be stopped before you start editing the configuration file iplir.conf and after you finish editing the file, you should restart it to apply the changes.

6.5.1 Configuring the "No external firewall" mode

To select this mode for a ViPNet workstation, specify the following parameters in the file iplir.conf:

- set the usefirewall parameter to off in the [id] section of the local node, i.e. external Firewall is not used.
- set the dynamic_proxy parameter to off in the [dynamic] section.
- if the host is a ViPNet coordinator, set the type parameter to internal for all interfaces that are used in the [adapter] sections.

Attention: usefirewall parameter had another value in previous versions, so at backup to old versions it's necessary to make settings for own workstation anew, using documentation of a version you want to install. Otherwise a logic of work of ViPNet Linux software will be broken.

6.5.2 Configuring the "ViPNet coordinator" mode

If a ViPNet coordinator is set up at the border of the LAN, hosts connected to it can select this computer as a node via which and on behalf of which they will exchange information with computers located in different networks.

The traffic from hosts working via the coordinator and intended for other hosts:

- located behind other network interfaces of the coordinator or
- not working via this coordinator and
- unavailable for broadcast packets

is automatically routed to the address of the coordinator (the IP and MAC addresses are translated), this coordinator translates the addresses and sends these packets further from its address. Traffic comes in other direction in a similar way.

The fact that encrypted packets are automatically routed from hosts to their coordinator makes it possible not to set it as the default gateway. This allows open packets to be routed to other gateways, if necessary, without specifying any additional settings on the computer after ViPNet is installed on it.

Automatic MAC address substitution can be done when using kernel patch only (see section 2.1). In case of using NETFILTER technology (see section 2.2), automatic MAC address substitution is impossible. In this case, you need to use a gateway by default. Coordinator or other router (in this network) that will redirect packets to coordinator can be used as a gateway.

The work of a host via a ViPNet coordinator in a large LAN divided into segments by all kinds of routers where allowed addresses and protocols are usually specified for security reasons is configured much easier because there is no need to specify a lot of allowed external addresses and protocols. Only UDP traffic with the internal addresses of its segment and the internal address of the ViPNet coordinator will circulate in this situation.

A host can be put behind a ViPNet coordinator only if one of networks interfaces on this coordinator is available for this host by its real address (i.e. there are no firewalls between them).

If this host is a ViPNet workstation, you should select its IP Addresses Server as a ViPNet coordinator working as a proxy server. If no IP Addresses Server is selected for this ViPNet workstation (it works without an IP Addresses Server) or the host is a coordinator, you can select any available server (see section 6.2).

To configure the operation of a host via a ViPNet coordinator, you should specify the following in the file iplir.conf:

- set the ip parameter in the [id] section of the ViPNet coordinator selected as a proxy server to any of its real IP addresses available for the local node if it is not yet

specified; specify the value of the port parameter in the same section, i.e. the destination port where packets for this ViPNet coordinator will be sent;

- if the local node is a ViPNet coordinator, set the type parameter to external in the [adapter] section corresponding to the network interface where the selected ViPNet coordinator is set up;
- set the usefirewall parameter to on in the [id] section of the local node;
- set the dynamic_proxy parameter to off in the [dynamic] section;
- set the proxyid parameter in the [id] section of the local node to the id value of the selected ViPNet coordinator;

After the host connects to the coordinator and if the latter is configured properly, the firewallip, port and proxyid values will be set in the [id] section of the selected ViPNet coordinator. Besides, the values of the firewallip and port parameters in the [id] section of the local node may change – they must correspond to the parameters of the selected coordinator.

6.5.3 Configuring the "Static NAT" mode

If the local area network has a firewall doing NAT where it is possible to configure static NAT rules providing the interaction with a certain internal network address via the UDP protocol using the specified port and it is necessary to interact with other computers that are external to the network firewall, you should select the ***"Static NAT"*** mode for the host.

In this case the IP address of local node and the port for accessing it must be firmly specified on the firewall. Besides, you should configure routing to the external firewall on the local node (the default gateway or routes for remote subnetworks).

For packets to be able to pass to a firewall (or a NAT device) where it is possible to configure static NAT rules, the following must be ensured:

- Outgoing UDP packets with the IP address and the port of the local node (source port) must be passed to any external address and port (with the source address substituted for external address of NAT device).
- Incoming UDP packets with the port of the local node (destination port) must be forwarded to the IP address of your host.

If there are several ViPNet hosts behind an external firewall of this type, each of them must be assigned its own access port.

To configure the operation of a host via a firewall with static NAT, you should specify the following in the file iplir.conf:

- set the usefirewall parameter to on in the [id] section of the local node;
- set the dynamic_proxy parameter to off in the [dynamic] section;
- if the local node is a ViPNet coordinator, set the type parameter to external in the [adapter] section corresponding to the network interface where the external firewall is set up;
- set the proxyid parameter in the [id] section of the local node to 0; if necessary, select the value of the port parameter from 1 to 65535 (by default, 55777). This parameter defines the port number from which packets transformed by ViPNet into the UDP format are sent from the local node (source port) and to which such packets come to the local node (destination port). It makes sense to change the port number only if there are several ViPNet hosts working via one firewall (or one NAT device) in your network. In this case all those hosts must have different port numbers.

In case the external firewall with NAT allows you to configure static rules only for incoming packets sent to the local node, i.e. pass only packets that have the destination address and port specified for them and forward them to the address of your host, you should set the fixfirewall parameter to on. In this case the IP address of the local node and the port for accessing it must be firmly specified on the firewall. They must be specified in the firewallip and port parameters in the [id] section of the local node.

If an external address of Firewall with NAT changes, you should set the fixfirewall parameter to off. In this case, IP address of access to your host will be registered by external packet parameters. In this mode the firewallip parameter is determined automatically according to information received from hosts located in the external network and it should not be modified manually.

The program sends information about the IP address of the firewall and about the access port to all other hosts the local node is connected to.

6.5.4 Configuring the "Dynamic NAT" mode

If the LAN is connected to a network via some NAT device where it is difficult to configure static NAT rules and it is necessary to interact with other hosts located in the network external to the firewall, you should select the ***"Dynamic NAT"*** mode for the host.

The mode of operation with this type of firewall is the most universal one and it can be used virtually in all cases. However, its main purpose is to provide a secure bidirectional connection between hosts working via NAT devices where it is difficult or impossible to configure static NAT rules. This situation is typical for simple NAT devices with minimum settings, for example, DSL modems, wireless devices and in other cases. It is also difficult to configure NAT devices located in ISP's offices (in home networks, GPRS and other networks where the ISP gives you a private IP address).

Let us give a brief description of how traffic goes through such NAT devices. All NAT devices automatically create the so-called dynamic NAT rules to pass UDP traffic. It mean that any outgoing packets are passed, their addresses and ports are translated, their parameters are recorded and dynamic rules for accepting incoming traffic are created based on these parameters. Incoming packets whose parameters correspond to the created rules are accepted within a certain time period (timeout). When this timeout expires after the last outgoing packet, the corresponding dynamic rules are deleted and incoming packets start being blocked. It means that it is impossible to initiate a connection with hosts working via such NAT devices from outside unless the corresponding outgoing traffic is detected.

To make it possible for a host working via such a NAT device and for all its ViPNet workstations (if the host is a ViPNet coordinator) to work in both directions, the host is configured to work via a firewall with dynamic NAT. At the same time there must be a coordinator permanently available in the external network (Figure 1). Let us call it the incoming connections coordinator (the forward_id parameter of the [dynamic] section). By default, it is the IP Addresses Server of the ViPNet workstation (if it is selected). After the host in the dynamic NAT mode connects to the network, it starts sending UDP packets to its incoming connections coordinator at the specified interval (25 seconds by default) (the timeout parameter of the [dynamic] section). The interval of sending these packets must not be much longer than the timeout of keeping a dynamic rule on the NAT device. Various NAT devices have different timeouts set for them, but usually it is not less than 30 seconds. It allows any external host to send a packet of any type to this host via its incoming connections coordinator at any time. In this mode the host always sends outgoing packets directly to their recipient bypassing its incoming connections coordinator. After the external host receives the first respond, it automatically starts sending all traffic directly to this host working via the NAT

device (Figure 1). This technology makes the host working via a NAT device always available (since dynamic rules are not deleted on the NAT device) and at the same time considerably increases the speed of exchanging information between the hosts.

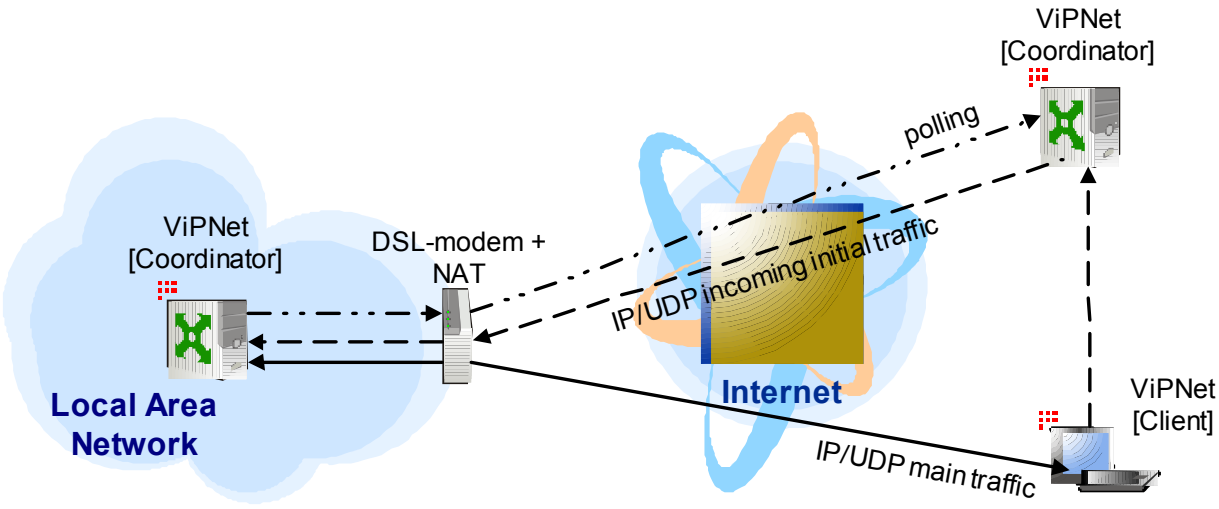


Figure 1 The diagram showing the operation of a host in the "Dynamic NAT" mode

If there are still problems with connections in the standard mode of operation via a firewall with dynamic NAT, it is possible to enable the option (the always_use_server parameter of the [dynamic] section) that allows any traffic to external hosts to be sent via the ViPNet coordinator. If you enable this option, all connections to other hosts will be established ONLY via the ViPNet coordinator selected for establishing outgoing connections, i.e. the technology described above (Figure 1) will not be used. In this mode the packet exchange rate may be reduced due to the longer route.

Note: in order to be able to interact with hosts working via some ViPNet coordinator, the host working in the ***"Dynamic NAT"*** mode must be linked to this coordinator.

To configure the work of the host in this mode, you should specify the following in the file iplir.conf:

- set the usefirewall parameter to on in the [id] section of the local node;
- set the dynamic_proxy parameter to on in the [dynamic] section;
- if the local node is a ViPNet coordinator, set the type parameter to external in the [adapter] section corresponding to the network interface where the external firewall is set up;
- select an external ViPNet coordinator to be used for establishing incoming connections (the forward_id parameter of the [dynamic] sections) if it is not selected yet. The value of this parameter for a ViPNet workstation always corresponds to the identifier of the selected IP Addresses Server, that is why if no IP Addresses Server is selected for a ViPNet workstation (the active parameter of the [servers] section is set to 0), it will be impossible for it to work in this mode. A record about this will be written to the system log. This parameter is specified manually for coordinators, but it also cannot be set to zero. Besides, the selected ViPNet coordinator must have at least one access address. Otherwise it will also be impossible to enable the mode being described.
- set the port parameter in the [id] section of the local node to a number from 1 to 65535 (usually, it is 55777), if it is not specified yet;
- change the values of the timeout and always_use_server parameters of the [dynamic] section if necessary.

In this mode the firewallip and port parameters of the [dynamic] section are determined automatically according to information received from hosts located in the external network and they should not be modified manually.

6.6 Configuring the work with a remote coordinator via a fixed alternative channel

According to basic work logic, every ViPNet node exchanges information with any other node via a certain access address and port, specified by firewallip parameters and an [id] section port. If the access address is changed, this data is sent to all other ViPNet nodes via service information exchange. It functions well, if there is only one possible access address for each node at every moment of time. Though it often happens that some nodes are connected to several independent channels, having several access addresses, each of which can be used to connect to other nodes. If you can't influence the choice of a channel, it may cause some problems, such as nodes would constantly switch channels, use an overloaded channel, etc.

To force the choice of a channel to connect different ViPNet coordinators, ViPNet Linux grants a special service, allowing the user to set a necessary channel for traffic exchange. For this purpose there is a [channels] section in the configuration file iplir.conf, within which you can define a set of necessary channels (see above) and register node groups (coordinators) for the defined channels. Group registration is optional when defining a channel name in the [channels] section. If you don't register any group, it'll be considered that no node groups are registered for this channel. If any node groups are defined, it means that they are registered to work with this channel. Though it doesn't mean that any node of the group would switch to this channel, as it is still necessary to define the external access IP-address and access port (optionally) for this channel. They are defined for each node of the group by channelfirewallip and channelport parameters correspondingly (you can find a detailed description of the syntax of these parameters above in this document). Thus, the values of channelfirewallip and channelport can by different for two different nodes in one and the same group. It allows fine tuning in a number of cases, e.g., when you need different ports to access different nodes via one and the same channel.

Therefore, every channel is identified by its own name (a unique identifier), node groups registered in it and access parameters, defined for each node in the group separately. The procedure of choosing a channel for a remote coordinator C can be described as following. If a channel A is defined in the [channels] section, a group B is registered in it (channel= A, B), a remote coordinator C belongs to this group (the corresponding parameter group= B is defined in the [id] section of the coordinator), and channelfirewallip value for this channel is defined in the [id] section of the coordinator C (channelfirewallip= A, IP), then connection with the coordinator C would always be done via the channel A with an external access IP-address IP. However, if there is a channelport parameter for the selected channel A (channelport= A, Port) besides the channelfirewallip parameter in the [id] section for the coordinator C, the value "Port" will always be used for an access port.

Remember that for the proper work with a remote coordinator via the selected channel symmetrical settings should be done on the remote coordinator, and the same channel for information exchange with your node should be configured. Otherwise, outgoing packets may be directed from your node to a remote coordinator via the selected channel, and the incoming packets may come via another channel.

If you choose the described above way of working with a remote coordinator via a fixed channel, the values of the firewallip and port (if channelport parameter has been defined) parameters in the [id] section of a coordinator will be self-changed after the startup of the control daemon to the values of the channelfirewallip and channelport parameters correspondingly. However, all the packets addressed to this coordinator will be sent via the selected channel to a

channelfirewallip access address and a channelport port, whether there is hard-wired connection, or not. There will be no auto switching to another channel.

To define another channel for information exchange with a remote coordinator, you should define the channelfirewallip and channelport parameters for another channel in the [id] section of the coordinator and define this channel in the [channels] section.

There are two ways of switching the coordinator between the channels:

1. The re-registration of the coordinator in another group registered for the work via the necessary channel, i.e. changing the value of the group parameter in the [id] section of the coordinator. This way is preferable if you need to switch only one node to another channel, and the other nodes will continue exchanging information via the earlier channel.
2. The re-registration of the group the coordinator belongs to via another channel, i.e. rewriting the group name as the value of another parameter channel of the [channels] section. This way is preferable if the node group switches to another channel, as the channel will be switched for all nodes of the group, with channelfirewallip parameter defined for the corresponding channel. Moreover, this way is also convenient if there is only one node in the group.

To stop working via a fixed channel, do one of the following:

1. Delete the group parameter from the [id] section of the corresponding coordinator. The node won't be registered in any group then. This way is convenient if you need to cancel working via a fixed channel temporarily, as to turn it on again you would just have to define the group parameter for this node anew.
2. Delete the corresponding group from the group list in the [channels] section of the corresponding coordinator. This way information exchange via fixed channels will be stopped for all nodes of the group.

You can find the examples of configuring selecting, switching and disabling alternative channels in **Appendix F. Examples of configuring the work of vipnet coordinators via fixed alternative channels**.

Attention! The procedure described in this section is meant to be used when no NAT devices or only NAT devices with static address translation are used, on a condition that a NAT device won't change its external address and sender port during its work. **We can't guarantee you the operability of this procedure if NAT devices with dynamic address translation are used.**

6.7 Configuring an Open Internet Server Coordinator

Let us consider the following problem. In a company due to security reasons it is forbidden for workstations in local network to access Internet resources (let us name it group A workstations). However there are some computers in local network which for some reasons require access to Internet (let us name it group B workstations). Using ViPNet technology, it is possible to create a dedicated virtual segment of local network workstations, which traffic will be completely isolated from the local network (and of course from other local workstations) when they are working in Internet.

To solve this problem:

- on the local network edge install a dedicated Open Internet Server coordinator;

- install on this coordinator an application proxy server (for example, Squid and others);
- install ViPNet Client on group B workstations.
- link group B workstations (with ViPNet Client installed) to the Open Internet Server coordinator.

You can use the following two scenarios of managing your local network:

1. Group B workstations are allowed working in Internet only. In this case, in ViPNet Manager, these hosts should be linked with the Open Internet Server Coordinator only. The first security level should be enabled on all ViPNet hosts. As a result, all Internet traffic coming through the application proxy server between group B workstations and coordinator (in other words - in the local network), will be packaged into secure envelopes, thus creating a protected VPN connection. There is no way for this traffic to go beyond the created virtual network segment. This scenario is equivalent to the isolation of physical local network segment and granting only this segment access to Internet.
2. Group B workstations are allowed to work in Internet as well as with other ViPNet hosts and not protected network hosts. This scenario is less secure than the first one. However, it provides a reasonably high level of protection against attacks for all workstations. The scenario objective is achieved by separating Internet activities from the ones with the local workstations. In this case, on a group B workstation:
   - when working in Internet, all local network traffic as well as traffic with other hosts except Open Internet Server coordinator is blocked.
   - when working in local network, all Internet traffic will be blocked.

   Such separation eliminates any threats that group B workstations can attack group A workstation online.

   For detailed instruction on how to configure ViPNet hosts to work with in Open Internet Server see ViPNet Client Administrator's Guide.

Open Internet Server coordinator:

- grants and controls access to public Internet resources using application level proxy servers;
- denies access to public Internet resources for group A workstations;
- creates a protected tunnel between itself and group B workstations, when these workstations work with public internet resources. This tunnel can not be accessed by other network hosts;
- is a Firewall that denies access to local network from public internet resources.

The ViPNet technology guarantees that no attacker from outside or inside can compromise network security and access any of local network hosts, which are forbidden to access Internet (group A workstations). If these workstations are under attack from Internet, only encrypted Internet traffic can arrive on these workstations. The encrypted traffic can not be processed by the operating system and will be discarded. Attacks from group B workstations are also impossible because all outgoing traffic from these workstations will be blocked except Open Internet Server traffic. At the same this configuration prevents unauthorized connection to the Internet.

To configure an Open Internet Server coordinator:

- Install an application proxy server (Squid, and others). **Note:** Proxy can be installed on the other computer tunneled by the Open Internet Server coordinator and placed behind a dedicated interface of this coordinator in the demilitarized zone (DMZ).

- Configure the security level settings as described below.

In the mode parameter, in the configuration file of the network interface, behind which local network workstations are located, specify the 1st security level (this security level blocks all unencrypted incoming and outgoing traffic).

The security level for the network interface, which routes Internet connections, depends on the proxy server installation:

1. If Proxy server installed on the Open Internet Server coordinator, in the network interface's configuration file, you can specify the 3rd security level (only single directed connections initiated by the proxy server are allowed)
2. If Proxy server is installed on the tunneled computer:
   - on network interfaces, which route internet connections, specify the 2nd security level.
   - in the firewall.conf file, in the [forward] table of the transit public network filters, add the following two rules:
     - the rule allowing single directed connection from all local network hosts to the IP address of the tunneled Proxy server;
     - the rule allowing single directed connection from Roxy server to all addresses behind the network interfaces routing Internet connections.

The second variant is preferable because the public Internet traffic does not reach Open Internet Server coordinator. Workstations, which access Internet resources, interact with this coordinator only through VPN and over service protocols, managed by ViPNet software. The re is no way to compromise the coordinator's security.

7 Working with ViPNet Linux

Nearly all commands you need to control ViPNet can be executed with the help of the iplir script located in the /sbin directory. The following commands are possible:

- iplir start – starts the ViPNet control program. If there are no configuration files, they are created in the same way as with the command iplir check.
- iplir stop – stops the ViPNet control program, the driver continues working.
- iplir unload – stops the ViPNet control program and unloads the ViPNet driver.
- iplir check – creates the configuration files and also checks their syntax if the files already exist. Besides, if the configuration files already exist, this command restores the filters in them necessary to pass ViPNet packets (broadcast packets on ports 2046, 2048, 2050 of the UDP protocol). These filters are enabled by default and they are necessary for the private network to function properly. However, if necessary, the user can disable them and in this case the command iplir check enables them again.
- iplir info – shows information about the local node or a remote one (see more details in documentation on console utilities).
- iplir view – allows you to view the traffic log of the local node or a remote one (see more details in documentation on console utilities).
- iplir passwd – allows you to change the ViPNet user password (see more details in documentation on console utilities).
- iplir config [command] – allows you to save, load, delete and view ViPNet configurations list (see section 9.6)

8 Failover System

8.1 Failover System Purpose

The Failover System is used to create the failsafe solution based on ViPNet Coordinator Linux. This system operates in two modes:

1. Standalone Coordinator
2. Hot Failover Cluster

When working in the Standalone Coordinator mode (used by default when installing the ViPNet Coordinator Linux) the failover system implements the following features allowing the ViPNet Coordinator Linux software services to work 24x7x365

- continuous services status monitoring and logging of system resources usage;
- service failure detection and unlimited number of subsequent attempts to recover the failed service;
- failure prevention during IP packets processing by the network protection driver drviplir;
- failure prevention of the failover system itself

Hot Failover Cluster mode is used for the one of the servers' with ViPNet Software functions hot replacement by another server if the first one fails. This document describes the standalone mode of the failover system. You can find more information on the Hot Failover Cluster mode in the special document Failover System Administrator Guide.

8.2 Failover System Components and Operation Principles

The Failover System consists of:

- a watchdog driver
- a daemon program failoverd running in the background mode.

Use the failover_info utility to request information about Failover System health and operation.

The watchdog driver operates on the low level and in most cases continues working even when the system stopped responding to any external events. During startup, failoverd daemon registers itself in the watchdog driver and periodically polls it to monitor the system health. If within the specified time interval the watchdog driver detects that no polling request has been made, it restarts the system. Before doing this it tries to write system cache buffer on the hard disk. However it is not always possible. If the failoverd daemon is stopped correctly (for example when modifying failover system configuration settings), it informs the watchdog driver about its stopping and the watchdog driver discontinues its monitoring of polling requests. As the result the system will not be restarted. Such a mechanism prevents the internal failures in the failoverd module.

During the OS start the failover daemon failoverd starts affiliated services and monitors them. The failoverd daemon constantly monitors the health of the following ViPNet services:

- ViPNet Control daemon (iplircfg);
- Transport module (MFTP) (mftpd);

During a service startup, the service is registered in the failover system and a notification period is set for it. The control over these services is implemented thereby. While working, the tracked service (application) detects its status and notifies the Failover system about it. If within the specified notification period the tracked application failed to notify the Failover system about its status or notified the Failover system about an internal failure, the Failover system considers such a behavior as an application failure and initiates the application recovery procedure. First the system tries to stop the failed application gracefully. If this attempt fails, the system forces the application to halt (incorrect stopping). Then the Failover system restarts the halted application.

While working the failoverd daemon logs all failures for each of the tracked applications including itself. If an application failed five times in a row, i.e. five attempts to correctly start the application were not successful, the Failover system deduces the applications' complete inoperability. In this case, depending on the Failover system settings (see 8.4 section) your operating system can be restarted or the failed application can be stopped without further monitoring.

Beginning from the 3.4 version VipNet Coordinator Linux failover system also monitors failures occurring in packets processing threads in the network protection driver drviplir (for more information seethe **Appendix B. ViPNet Linux kernel modules** chapter). To accomplish this, while starting, both the watcher and tracked applications generate a special request to the drviplir driver. If an error code flagging that one of the driver's packet processing threads failed has been returned in response to this request, the tracked application informs the watcher application about this failure. The watcher application then proceeds according to the logic described in the paragraphs above.

The control daemon iplircfg periodically polls drviplir driver (requesting information about packets log). The logic of finding failures in the packets processing threads is the same as the one of the tracked application start.

Because of the described mechanism it takes the Failover system seconds (or minutes, depending on the overall computer performance and some external factors) to trace driver failures and immediately take corrective actions (reboot a computer if settings enable this).However this mechanism is unable to trace all possible drviplir driver failures, for example the hang of one processing threads, etc. Therefore, we do not recommend to consider this mechanism as universal solution for all driver-level issues and failures.

If administrator has correctly stopped the tracked application executing one of the appropriate commands (iplir stop or mftp stop) it will be unregistered in the failover system and its monitoring will be discontinued. In this case to continue working with the application the administrator should manually start the application (by executing the iplir start or mftp start commands).

If during the failoverd daemon startup it became known that some of the tracked daemons were stopped manually, the warning message will be created in the syslog. This message will also will be displayed on the terminal (if existed) .The same warning will also be created when 10 consecutive checks of the same daemon show that it was stopped manually.

8.3 Managing Failover System

When working in standalone mode administrator's actions targeted at failover system management are reduced to a minimum: the administrator can start or stop failoverd daemon. To perform these actions, use the controlling failover script , which is placed in the in /sbin directory during the installation process.

To start the failover system, execute the failover start command. The failoverd daemon starts thereby. To stop the failover system, execute the failover stop command. The daemon will stop correctly and notify the driver about this event.

During the system startup the watchdog driver is loaded automatically and the failover start command is executed. As a result the failoverd daemon will be loaded, which in its turn starts other tracked ViPNet Coordinator Linux (iplircfg, mftpd) services.

Warning: If you execute the failover start command manually in the standalone mode, tracked applications will not be reloaded automatically.

Execute failover info command to view information about failover system health and operation status (see 9.3 section).

8.4 Configuring Failover System

To configure failover system parameters the administrator should edit the failover.ini file located in the /ets directory.\

Several sections comprise the failover system configuration file. Each section begins with a line containing the section name in the square brackets. Each section contains several parameters. The line with parameter begins with the parameter name, then the **'='** symbol goes, after it - the space and at last the value of the parameter. To configure the failover system in standalone mode, use the following parameters:

- The [misc] section contains parameter, responsible for the system action when complete inoperability of any tracked application has been detected.

  * reboot – instructs the failover system to reboot the OS if any of tracked application is not able to start correctly (see section 8.2). This parameter accepts yes or no values. The value yes enables OS restart, the value no – disables the OS restart. This parameter is required.

    When installing over the earlier ViPNet Coordinator Linux versions (earlier than 2.8.18) the reboot parameter is defined automatically depending on the usewatchdog parameter of the [watchdog] section (this section is deprecated and will be removed after the current ViPNet Coordinator Linux version installation). If the watchdog parameter was enabled (had the on value) then yes will be chosen for reboot parameter, in other cases – no.

- [debug] contains parameters defining failoverd troubleshooting log options. (For more information refer to 13 section). It includes the following parameters:

  * debuglevel – the logging level, the number from -1 up to 5, by default 3. The **-1** value disabled logging.

  * debuglogfile – is an identifier, defining log location. The format of this identifier is the following: **<protocol specifier>**, **<URL specifier for this protocol>**. See the 13 section for the detailed description of possible values of this parameter. The default parameter value is /var/log/iplircfg.debug.log, that corresponds to logging to a specified file.

Before you start editing the configuration file failover.ini, execute the failover stop command to stop the failoverd daemon. Upon finishing, execute the failover start command to run the failoverd daemon again

9 Console utilities

9.1 Event log Information Viewer

The dbviewer utility is used to view the event log created by the driver control program. When you are working with the viewer, you can filter events by the following criteria:

- date intervals;
- unprotected workstation addresses;
- ViPNet host names;
- packet direction – incoming or outgoing;
- a range of local port numbers or one local port number for tcp, udp;
- a range of remote port numbers or one remote port number for tcp, udp;
- IP protocol;
- the network interface a particular packet was processed by;
- event type.

Upon installation the dbviewer utility is placed in the /sbin directory. Usually it is started from the iplir script using the command iplir view. Once started this way, the program reads the configuration file /etc/iplirnetpsw that must have the following format: the first line contains the access password to the computer whose event log is to be viewed, the second line contains the domain name or the IP address of this computer. The rest of lines are ignored. After the installation the default address of the computer is set to 127.0.0.1.This means that the log of the local computer is supposed to be viewed. You should enter the password manually.

The program allows you to view the log on the local computer as well as any specified ViPNet host, which makes it easier to control the network from one place. The IP address of the computer for the program to connect to in order to view the event log can be specified in the file /etc/iplirnetpsw. Besides, you can create another configuration file, enter the address of the necessary computer and password to it into it and pass the name of this file as a parameter to the dbviewer program while starting it. For example: /sbin/dbviewer /etc/myconf /etc/iplirpsw.

After you start the viewer, it tries to connect to the specified computer. For the connection to be established, the computer must be available on the TCP/IP level, the control program must be running on it and the specified access password must be correct. If the connection attempt fails, the program informs you about it and finishes.

If the connection is successfully established, the terminal screen will look as follows:

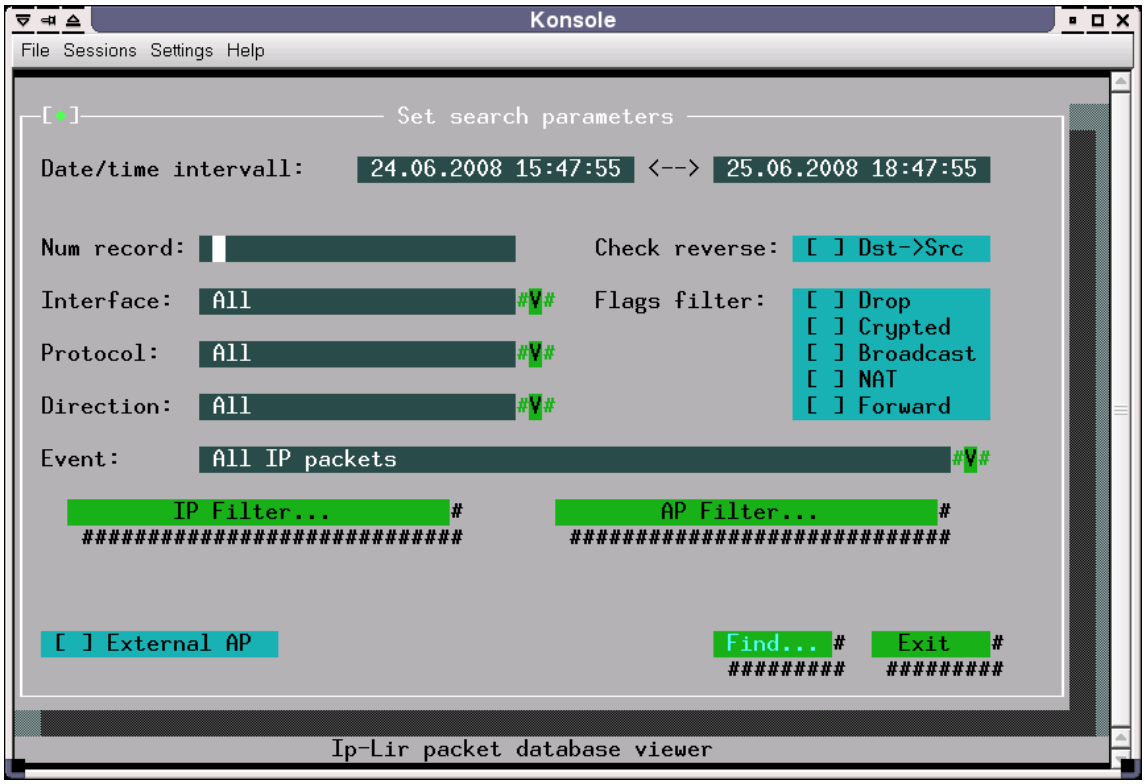


Figure 2 Specifying the parameters for searching the event log.

Let us describe the main elements of the window (Figure 2):

- *Date/Time interval* – the date and time period to search for events about registering packets in the following format: DD.MM.YYYY HH:MM:SS;
- *Num record* – the number of records in the reply to the request;
- *Interface* – you should select a network interface from the list of available ones. An interface is considered available, if it has a packet log. Search for packets will be performed in the log of the selected interface. For the value of this parameter, you may specify an interface name or *All* to work with all interfaces;
- *Protocol* – the name of the protocol to search only packets belonging to this IP protocol. You can specify either the name of the protocol or *All* to search all protocols.
- *Direction* specifies a direction of packet travel in the required events;

  All – all incoming and outgoing packets

  Incoming – incoming packets;

  Outgoing – outgoing packets;

- *Event* specifies an event type or class for searching for events; the required type or class is selected from a list; by default, the search is performed among all types of events.
- *IP Filter* displays a dialog box for specifying the following query parameters:
  - *Source IP address* – All, a range or a single value for the allowed source IP address;
  - *Destination IP address* – All, a range or a single value for the allowed destination IP address;

- *Source port* – a range or a single value for the allowed source port number (0-65535) for tcp and udp protocols;

- *Destination port* – a range or a single value for the allowed destination port number (0-65535) for tcp and udp protocols;

- *AP Filter* displays a dialog box, where you can narrow your search by specifying the sender's and/or recipient's host: Source AP and/or Destination AP (see the Figure 3).

- *Event* – specifies the type and class of events for searching. The necessary type and class are selected from the list. By default, events of all types are searched.

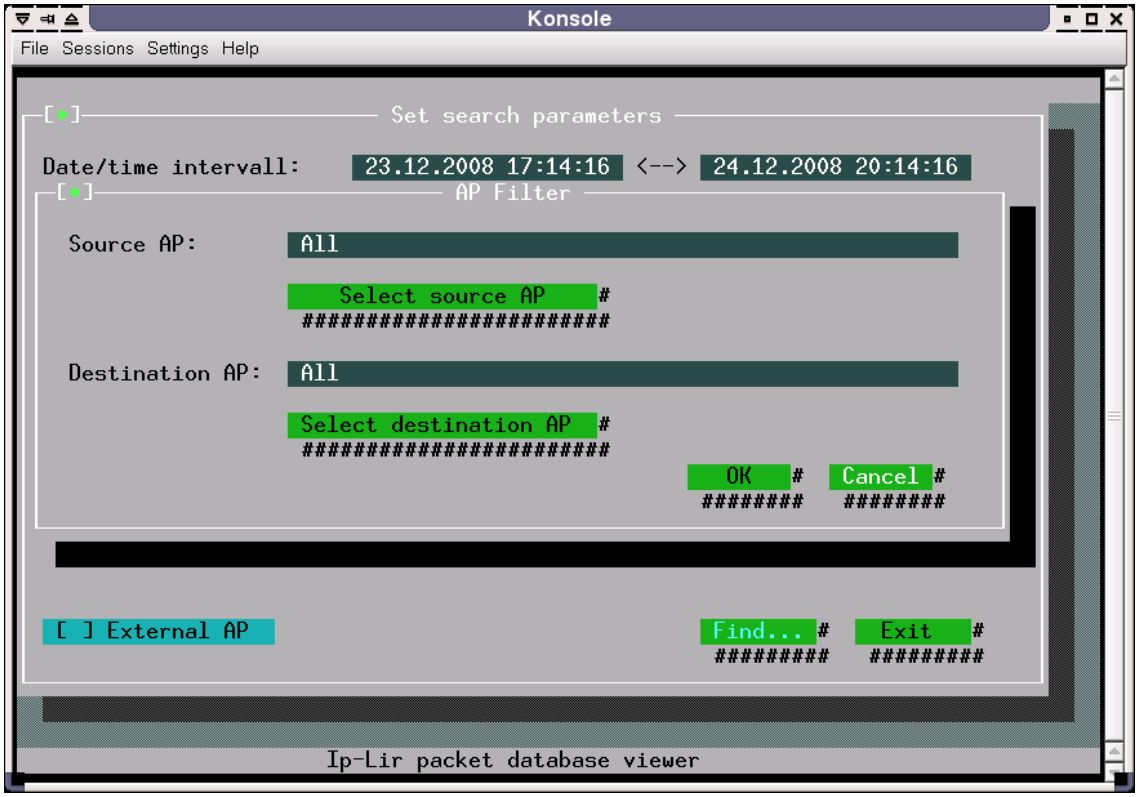


Figure 3. Specifying Source AP and/or Destination AP host

To choose the sender's and/or recipient's click **Select source AP** or **Select destination AP** respectively. When you do this, the window displaying the searchable list of protected hosts will be shown (see the Figure 4).

Figure 4. The protected hosts list to choose sender or recepient from

This window displays the list of all protected ViPNet hosts linked with the current coordinator. All hosts in the list are sorted in the alphabetical order. The hexadecimal host identifier is displayed in the left column. Apart from the hosts the list contains the **All** element, by choosing which you will include all available hosts in your search.

Click **Search** to start searching. Another window will display the search substring allowing you to manually type or select from the drop-down list previously typed substring. You can use both host name and unique host identifier as your search criteria. This means, that the search will find out whether the typed substring matches the value both in the **Name** column and in the **ID** column.

Click **Search next** to find next element matching the previously selected search substring quickly.

To start searching by the specified criteria, you should click the **Find** button located at the bottom of the main window. To exit the program, click **Exit**.

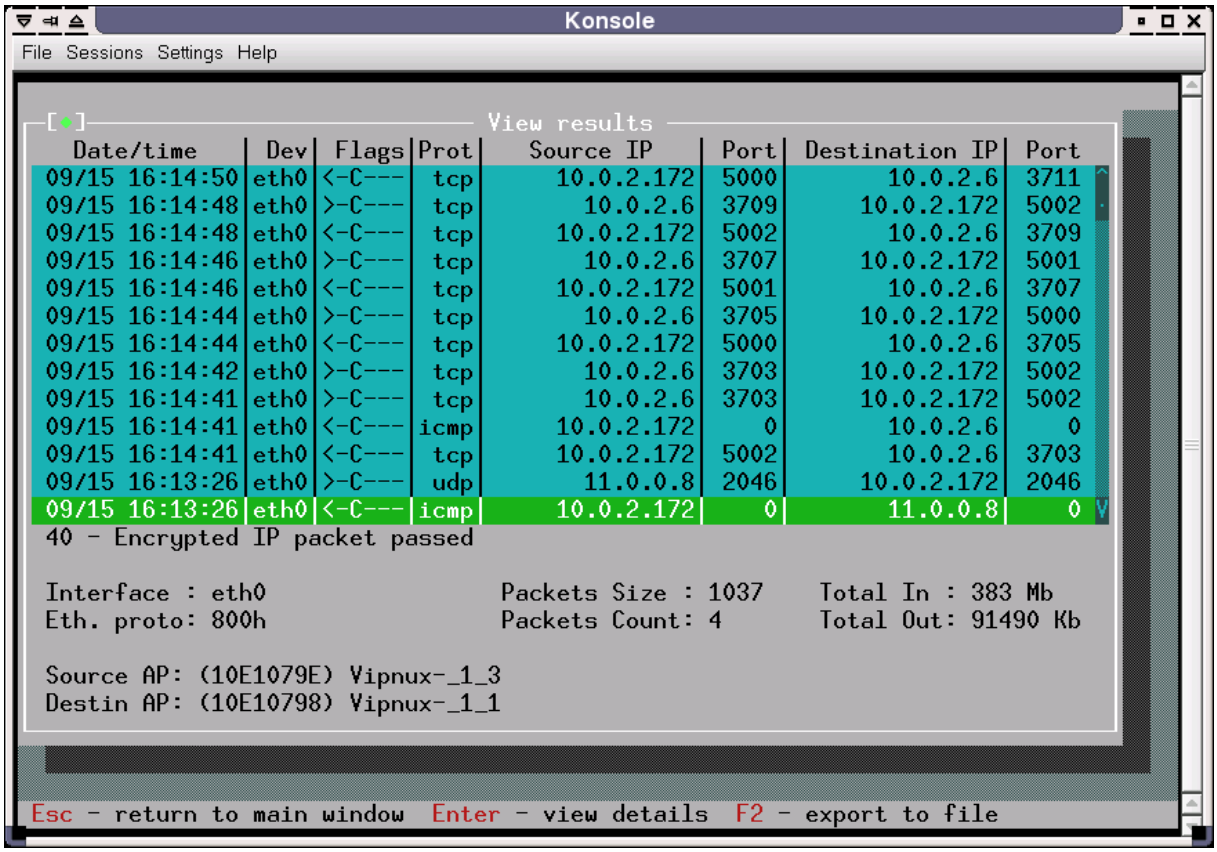


Figure 5. Displaying the list of found records.

After the search by the specified criteria is over, information is displayed in the "View results" window (Figure 5). The entries are sorted by the packet registration time. The list consists of the following columns:

- Event registration date and time;
- Interface the event was registered at;
- The direction of the logged packet: "<" – outgoing, ">" – incoming and event flags combination:
  - **'C'** – encrypted packet;
  - **'B'** – broadcast packet;
  - **'D'** –blocked packet;
  - **'T'** – transit (routable)packet;
  - **'R'** – the packet will be processed by the public network NAT rules;
  - **'N'** – the packed has been processed by the public network NAT rules.
- Protocol
- Source IP address;
- The local port for the tcp and udp protocols;
- Destination IP address;
- The port on the remote host for the tcp and udp protocols;

The lower part of the window displays information for the selected event:

Event name;

Interface;

Protocol;

- Packet size. The total size of all packets belonging to this event (if the counter is larger than 1) is displayed. The full size of encrypted packets with all services headers necessary for working in the private network is displayed.
- Number of packets related to the event.

Source AP;

Destination AP;

You can view more detailed information about any event selected in the list (Figure 6). To do it, you should click [Enter].

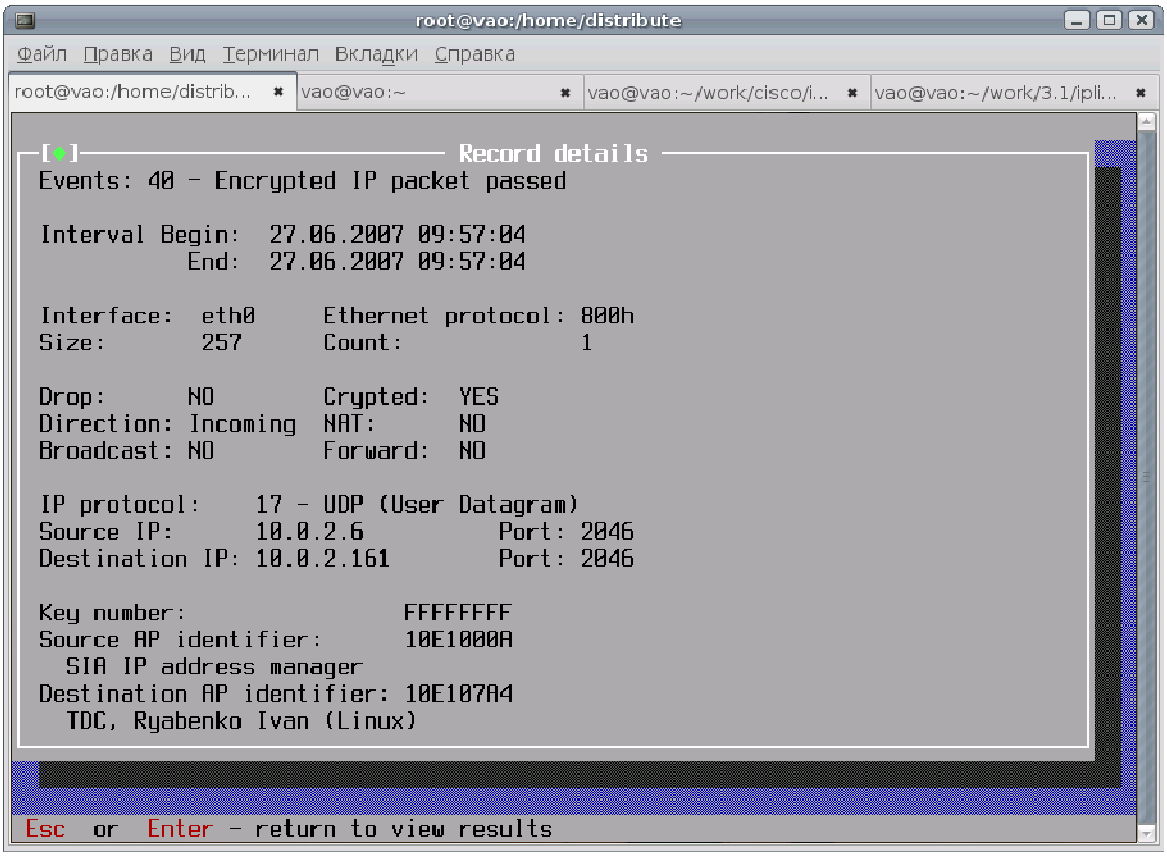


Figure 6. Detailed information about the event.

You can also export the displayed list of packets into a text file if you click F2 button. After that you can either select the file you want to export the log to or enter its name manually. All fields for each event displayed in the View Results window are saved to this file.

Information about packet size is supported from version 2.8-157. If you install this version or later above an earlier version, packet logs will be converted into the new format, but information about the size of packets logged by the old version will be unavailable and displayed as N/A (Not available). When you are viewing a remote log created by the old version,

information about the size of packets is also neither available nor displayed. The same happens when you use the old version to view a remote log created by the new version.

The **TotalIn** and **TotalOut** boxes display the total size of all incoming and outgoing packets. You can find these boxes at the bottom of the window near the list of found entries (Figure 5). The total size is calculated for all packets returned by the query. If a returned entry does not contain information about packet's size, these entries will be excluded from the total size calculation. In this case the asterisk in the **TotalIn** and/or **TotalOut** boxes will be placed after the calculated sum. This asterisk flags that not all traffic data have been calculated.

When displaying total size, the following measurement units are used:

- If the total size is less than 100 kilobytes, the size will be displayed in bytes; the "b" suffix is added to the size displayed in the box;
- If the total size is greater than 100 kilobytes but less than 100 megabytes, the size will be displayed in kilobytes; the "Kb" suffix is added to the size displayed in the box;
- If the total size is greater than 100 megabytes, the size will be displayed in megabytes; the "Mb" suffix is added to the size displayed in the box.

9.2 Remote User Information Viewer

The program is used to view system and statistical information received from a remote ViPNet host. System information includes the version of the control program and the driver installed on the computer, the identifier and name of the remote ViPNet host, the security level for each interface, etc. Statistical information includes information about processing incoming and outgoing IP packets for the specified interface.

The program is named iplir_remote_info and located in the /sbin directory. Usually it is started from the iplir script with the commands iplir info and iplir ifinfo. Once started this way, the program reads the configuration file /etc/iplirnetpsw that must have the following format: the first line contains the access password to the computer whose event log should be viewed, the second line contains the domain name or the IP address of this computer. The rest of lines are ignored. After the installation the default address of the computer is set to 127.0.0.1. That is, information about the local computer is supposed to be viewed. The password must be entered manually.

Once the command iplir info is executed, the program displays system and statistical information for the first network interface in the system. The following statistical information is displayed:

- *Not crypted packets passed* – the number of received and sent unencrypted packets;
- *Not crypted packets dropped* – the number of blocked incoming and outgoing unencrypted packets;
- *Crypted packets passed* – the number of received decrypted packets and sent encrypted packets;
- *Crypted packets dropped* – the number of blocked incoming and outgoing encrypted packets;
- *Not crypted broadcast packets passed* – the number of received and sent broadcast packets;
- *Not crypted broadcast packets dropped* – the number of blocked incoming and outgoing broadcast packets;

- *Crypted broadcast packets passed* – the number of received and sent encrypted broadcast packets;
- *Crypted broadcast packets dropped* – the number of blocked incoming and outgoing encrypted broadcast packets.

To view information about other network interfaces (in case there are several network adapters installed on the computer), you should use the command iplir ifinfo. First, you should use the command iplir info that shows the list of available interfaces and then you should specify the name of the necessary interface in the command iplir ifinfo. For example: iplir ifinfo eth1.

You can also start the program iplir_remote_info directly. The command has the following format:

iplir_remote_info <parameters> netconfig_file /etc/iplirpsw,

with the following possible parameters:

-s – show system information

-i <interface> - shows statistics for the specified interface

-d <time> - the frequency of updating the information

netconfig_file – it is either /etc/iplirnetpsw or another file of the same format.

For example, you can have several configuration files for different computers and specify the corresponding file to view information about a particular computer.

If the update frequency is specified, the program does not finish immediately, but updates the information with the specified frequency and displays the updated information. You can interrupt the operation of the program in this mode by pressing Ctrl+C or sending it the TERM or INT signal.

9.3 Failover System State Information Viewer

Failover system information view utility is used for reading the current information of failover system sate (refer to 8 section). Information includes:

- ViPNet Software and failoverd versions, installed on the station.
- An identifier and a name of the station.
- Failover system working mode (single or hot failover cluster).
- Station local time.
- Current information on services, controlled by failover system (see below).

The utility is called failover_info and after the installation located in /sbin directory. Usually it lunches from controlling failover script by the failover_info command. When started in such a way the utility reads the /ets/iplimetpsw configuration file. When using the failover_info command the utility writes necessary information on console.

When working in single mode the information format is following:

Versions: ViPNet 3.2 (475), daemon 1.3 (14)
Workstation configured for ID 10E1079E (Vipnux-_1_3)
The workstation works in a single mode of protection against failures
Workstation time (utc: 1174558030) Thu Mar 22 13:07:10 2008

*failover mode * single* – the mode of the failover system working;
*failover uptime * 6d 0:23* – failoverd working time;

total cpu * *0%* - general CPU consumption;
failover state * *works* – failoverd state;
failover cpu * *0%* - CPU consumption by failoverd;
iplir state * *works* –controlling iplircfg daemon state;
iplir cpu * *0%* - CPU consumption by iplircfg daemon;
mftp state * *stopped* – mftpd daemon state;
mftp cpu * *0%* - CPU consumption by mftpd;

Mode marks:

- *single* – single mode;
- *active* – servers hot failover cluster "active" mode;
- *passive* – servers hot failover cluster "passive" mode;

State marks:

- *works* – application is working correctly (in the view of failover system);
- *stopped* – application is stopped by user;
- *unknown* – state is unknown. This state can be determined in the case of the application failure was identified, the attempt to stop the application was done but there is no any information on the application start;

The format of writing the information in the hot failover cluster mode is described in "ViPNet Failover System Administrator Guide".

9.4 User Password Changer

The program is used to change the user password in ViPNet. Since passwords are initially set by the ViPNet administrator, the user may need to change his password after the installation.

The program is called iplirpasswd and is place in the /sbin directory after installation. It is usually started from the iplir control script using the iplir passwd command. After this the user is prompted to enter a new password in the symbol screening mode, i.e., password symbols are not displayed during entry. If the password is changed successfully, the program terminates, displaying the corresponding message to the user.

Before changing the password, you should stop ViPNet SW service (the iplircfg control daemon, the failoverd failover daemon, the mftpd transport module), and make sure that the /etc/iplirpsw file contains the old password. After changing the password, you should enter the new password in /etc/iplirpsw and /etc/iplirnetpsw files, and start the SW ViPNet stopped previously.

9.5 Key Database Distribution Package Unmerger

The unmerge program is used to unmerge the ViPNet distribution key set. The key set given to the administrator of each Linux station by the ViPNet administrator is a file with the .dst extension that should be later unmerged with the help of the unmerge utility. After the installation this utility is saved to the /sbin directory.

The unmerge program takes two parameters when it is started:

unmerge <distribution_file> <unmerge_directory>

The unmerge program unmerges the specified file into the specified directory. If the directory doesn't exist, it will be created. If the directory already exists and is not empty, the corresponding message will be displayed to the user, asking for a confirmation to continue. If the user agrees, then before unpacking, all data in the specified directory is completely deleted. If the user refuses, the program terminates. If the user has specified the optional –f parameter, then the directory cleanup confirmation if not requested – the program automatically deletes all data from the directory.

9.6 ViPNet Configuration Program

Currently, information in the configuration files, host and coordinator links and key sets of ViPNet Linux is not protected against being accidentally damaged due to software bugs or users' incorrect actions. In particular, it may sometimes lead to a situation when configuration files get damaged after the key set and/or host and coordinator links are updated and the system cannot work normally. To avoid such situations, ViPNet Linux has a special feature allowing you to save configuration files, host links and key sets (hereinafter configuration).

Using this feature you can save the current configuration, load one of the previously saved configurations or delete them. Moreover, the current configuration can be saved automatically (an autosaved configuration) before any updates are made depending on the settings (see ViPNet Linux MFTP Administrator's Manual). A previously saved user configuration can be loaded only manually with a command executed by the user. Before loading you will be prompted to save the current configuration for future use, because if not saved, the current configuration will be lost. Previously saved configurations can be deleted only manually as well.

There are two configuration types:

- Partial configuration. It includes all configuration files of ViPNet components, including iplir.conf, mftp.conf, failover.ini, etc.
- Full configuration. It includes all files of a partial configuration and also all ViPNet host links and key sets. Only temporary service files, log files and MFTP message queue are not included in a full configuration.

The ViPNet Linux software includes a special utility for managing configurations named /sbin/iplir-config-manager. It is started from the iplir script also located in the /sbin directory. The following set of commands for working with configurations is supported:

- iplir config list VERSION– allows you to view the current set of saved configurations in the following format:
  - Unique configuration name,
  - ViPNet software version for creating this configuration (if presents);
  - configuration type,
  - saved on,
  - loaded on

  For example:

  "Config 1", full, saved on 21.03.2005 at 11:49, never loaded

  "Config 2", part, saved on 21.03.2005 at 11:49, never loaded

  "autosave-2005-03-24-17-05-31", full, saved on 24.03.2005 at 17:05, loaded at 01.04.2005 at 12:45

  "rollback-2006-10-12", version 2.8.16, part, saved on 12.10.2006 at 11:30, never loaded

  In case of a configuration has been automatically saved, its name starts with “autosave”. In case of rollback configuration, a name of the configuration begins from “rollback” (see chapter 5).

  VERSION – optional parameter, specified in the following format Major.Minor.Subminor. This parameter allows filtering configurations by ViPNet version, which this configuration

is related to. If this parameter presents, only information about configurations with version number less or equal to VERSION value, will be output to the console.

- iplir config save NAME TYPE – saves the current configuration of the specified type. NAME – the name to save the configuration under.

  TYPE – configuration type (full or partial, the default is full).

  If the configuration with the specified name already exists, the iplir config utility asks the user whether the existing configuration should be overwritten.

- iplir config load NAME VERSION – loads the configuration with the specified NAME and VERSION. Before restoring the configuration, the iplir config utility asks the user if the current configuration should be saved. It is recommended to agree to it and enter the name to save the current configuration under. If the user refuses to save the current configuration, the program asks him to enter a special phrase to avoid errors. For example:

  Save current configuration [Y/n]? n

  You are about to overwrite your existing configuration.

  This is unsafe. To continue, type

  >>> Yes, do as I say

  in response to the prompt:

  >>>

  In this case the user has to enter the phrase "Yes, do as I say" to continue the operation. To cancel the operation, just press Enter.

  VERSION – a parameter, specifying software version, which configuration is related to. The parameter is specified in the following format: Major.Minor.Subminor. If the parameter is not specified, the command will be related to kept configurations only without information about software version. If there are several configurations with identical names, but with different software versions in the database, that, in order to identify a configuration you want to restore this parameter needs to be specified, otherwise the utility will give a corresponding message about error.

  At trying of restoring a configuration, the iplir config utility compares current version with a version, which the configuration is related to. If current version is more than a version, which the configuration is related to or equal to it, restoring will be done without additional confirmation. Otherwise, a user will be informed that the configuration is related to more new software version, and, probably, configuration files need to be edited manually for correct acceptance (see chapter 5). Then the user will be asked about restoring or no this configuration. At negative answer the configuration won't be restored.

- iplir config delete NAME VERSION – deletes the previously saved configuration with the specified NAME. Wildcards, such as "*" (any number of symbols) and "?" (one symbol, any), can be used as a name. So, it's possible to delete configurations using a mask. For example:

  iplir config delete "Config *"

  iplir config delete "autosave-2005-08-??-*" 2.8.16

  If more than one configuration can be found with using specified template, a user will be notified about it and a confirmation of such a removing will be asked.

  When using masks, you should always put wildcards in double quotes (see above). Otherwise the command can be incorrectly interpreted by the operating system's command shell thus resulting in improper system functioning.

VERSION – a parameter, specifying software version, which configuration is related to. The parameter is specified in the following format: Major.Minor.Subminor. If the parameter is not specified, the command will be related to kept configurations only without information about software version. In case of parameter VERSION is not specified, but parameter NAME is specified as mask (substitution symbols are used), all configurations corresponding to this mask irrespective of software version will be deleted.

All ViPNet services should be stopped before you save and load configurations. If this requirement is not met, the corresponding message will be displayed to you by iplir script and the command will no be executed.

10 Fine-tuning the ViPNet Linux software

This section is intended only for qualified system administrators who may need to configure ViPNet for working under various nonstandard conditions.

ViPNet includes three drivers

- drviplir – main network driver;
- itcswd – watchdog driver;
- itcscrpt – crypto-driver

In some cases described below it may be necessary for some of these drivers to work in a mode different from the default one. It is possible if you pass parameters to a driver with the insmod command when it is being loaded. The name of a parameter passed to a driver looks as follows: <driver_name>_<parameter name>. The value of the parameter comes after its name and the '=' sign.

In order to load ViPNet drivers with parameters, you should add the necessary parameter to the variable containing the set of parameters of the corresponding driver. These variables are defined in the file /etc/iplirSKnums. Thus, the parameters of the drviplir driver are defined in the DRVIPLIR_PARAMS variable, the parameters of the itcswd driver are specified in the ITCSWD_PARAMS variable, etc.

10.1 Changing major device numbers used by drivers

All ViPNet drivers use dynamically assigned major device numbers. After they get a dynamic number, they create a special file (a device node) that application programs use to address the driver. Thus, application programs do not depend on the major device number used by the driver.

But when ViPNet drivers are used together with other drivers that do not use the mechanism of dynamically assigning major device numbers, it may happen that the number dynamically assigned to a driver coincides with the number used by another driver. In this situation it is possible to give ViPNet drivers static device numbers that they will use. The <driver_name>_major parameter is used to assign static device numbers. The value of this parameter must be the major number of the device that the driver should use. For example, if we want the drviplir driver to use number 250, we should add the following record to the DRVIPLIR_PARAMS variable: drviplir_major=250. Of course, each ViPNet driver must be given a different number.

10.2 Fragmentation of Encrypted ViPNet Packets

In the previous versions, the ViPNet driver reduced the MTU size for all interfaces processed by ViPNet. It was done to avoid packets fragmentation because of adding ViPNet service information.

Starting with version 2.8-284, ViPNet driver uses new MSS control technology for TCP connections. This technology allows to avoid packets fragmentation without changing MTU on interface. Therefore parameter drviplir_mtudiff is not used by driver. At update of old ViPNet version this parameter will be deleted from file /etc/iplirSKnums!

10.3 Setting the maximum size of a packet queue in the driver.

When the ViPNet driver processes network packets, it uses a special queue where all incoming and outgoing packets from all network interfaces are placed. As packets are being processed, the queue can either grow or shrink depending on the traffic, the number of interfaces, computer performance rate, etc. In order to reduce the excessive use of RAM, which leads to the

unstable work of the system, the size of the queue is limited to 2048 by default. It means that if the queue already contains the maximum number of packets, the rest of incoming/outgoing packets are blocked until there is a vacant place in the queue. When the size of the queue reaches 90% of its maximum size, the driver displays a special message to inform the user about it. In case with typical network traffic, this situation is very unlikely to occur. But in some special cases (for example, if there is a lot of UDP traffic) the queue may overflow. The drviplir_pqsize parameter is used to avoid such situations. It allows you to change the size limit for the driver queue. This parameter should be added to the DRVIPLIR_PARAMS variable of the file /etc/iplirSKnums. For example: drviplir_pqsize=4096. In this case the queue size limit will become 4096.

10.4 Linux System Date and Time Setting when Using ViPNet Software

If during the operation of the ViPNet SW the Linux station administrator need to change system time, then they ABSOLUTELY need to stop the following ViPNet SW services before that: the failover daemon (failoverd), the control daemon (iplircfg), and the transport module (mftpd), and, after setting new time, start them again manually

. These daemons use current system time during their work, therefore, if you want to change the system time during their work, a logic of functioning of different subsystems may be broken. In this situation, changing system time in back direction is more dangerous.

10.5 Manually reassigning virtual node addresses

In the standard operating mode, as described in section 6.2.1, the control daemon automatically assigns virtual addresses to new ViPNet nodes added to links. During this, virtual addresses for already existing nodes are not changed. When a node is deleted from a line, the corresponding virtual address is considered unused, and a "hole" in the current virtual address range is created. In some cases this algorithm is ineffective, e.g., if, for some reason, you need to use a limited virtual address range. In this case the user needs a mechanism that will allows to reassign virtual addresses to nodes, filling the "holes" created earlier when removing links to nodes.

To reassign virtual node addresses, you may use the startvirtualiphash parameter (see section. 6.2), which contains a has of the starting virtual address (the startvirtualip parameter). Using this parameter, when the control daemon is started, it determines whether the starting virtual address is changes, and, if this is so, reassigns virtual addresses for all nodes. To reassign virtual addresses without changing the starting virtual address, you should stop the control daemon, delete the line containing the startvirtualiphash parameter, and restart the daemon. In this case, all virtual addresses for all ViPNet nodes starting from the address specified in the startvirtualip parameter will be reassigned. All "holes" will be filled during this..

11 Retrieving information about a ViPNet host using the SNMP protocol

ViPNet Linux can work with an SNMP server and makes it possible to obtain information about the uptime of a remote system, its interfaces and their modes, the number of packets passed and blocked, etc. as well as to notify a remote network management station (NMS) about the most important events in the system.

Currently, ViPNet Linux works only with the ucd-snmp server, which is distributed freely and comes together with ViPNet Linux. The source code of this server and also all files necessary for the SNMP system to function can be found in the snmp subdirectory of the ViPNet Linux software distribution package.

Before installing this SNMP server, make sure there is no other SNMP server installed on the system. After that you should execute the install-snmp script also located in the snmp

directory. This script automatically compiles and installs the SNMP server together with modules necessary to work with ViPNet Linux. For the compilation to be successful, all software packages listed in the «Installing ViPNet Linux» section must be installed on the computer. During the compilation process the user is asked questions about the administrator's e-mail address, system location, etc. They should be answered properly.

If you plan to use notifications (SNMP Traps), after the installation you should create (if it does not exist) and edit the file /usr/local/share/snmp/snmpd.conf by adding the following line to it:

trap2sink <address>

where <address> is the IP address of the computer that will receive notifications.

After that you should start the SNMP server with the command /usr/local/sbin/snmpd. If you need the server to be started each time the system starts, you should do it yourself by adding it to one of the startup scripts.

Special network management software (NMS) is used to receive information via the SNMP protocol, e.g. HP OpenView. Before you start working with a ViPNet host, you should import a special ViPNet MIB file into NMS. This file describes SNMP objects used by ViPNet. It is named VIPNET-MIB.txt and located in the snmp directory. What you should do to import this file depends on the NMS you use and it must be described in its documentation.

After that, you can use this NMS to get information about a ViPNet Linux host. ViPNet Linux uses the branch .iso.org.dod.internet.private.enterprises.infotecs.vipnet (.1.3.6.1.4.1.10812.1) for that.

ViPNet Linux uses the following objects (data for all objects is read-only):

- .1.3.6.1.4.1.10812.1.1.1 – the ViPNet network identifier of this host
- .1.3.6.1.4.1.10812.1.1.2 – the name of the ViPNet host
- .1.3.6.1.4.1.10812.1.1.3 – the name of the user whose password was entered to start ViPNet
- .1.3.6.1.4.1.10812.1.1.4 – system uptime
- .1.3.6.1.4.1.10812.1.2.1 – the number of network interfaces in the system.
- .1.3.6.1.4.1.10812.1.2.2 – the sequence of objects describing network interfaces.
- .1.3.6.1.4.1.10812.1.2.2.1 – each of the objects describing network interfaces. It contains the following fields:
  - .1.3.6.1.4.1.10812.1.2.2.1.1 – interface number
  - .1.3.6.1.4.1.10812.1.2.2.1.2 – interface system name
  - .1.3.6.1.4.1.10812.1.2.2.1.3 – the security level of the interface in ViPNet
  - .1.3.6.1.4.1.10812.1.2.2.1.4 – the number of passed unencrypted incoming packets
  - .1.3.6.1.4.1.10812.1.2.2.1.5 – the number of blocked unencrypted incoming packets
  - .1.3.6.1.4.1.10812.1.2.2.1.6 – the number of passed unencrypted outgoing packets
  - .1.3.6.1.4.1.10812.1.2.2.1.7 – the number of blocked unencrypted outgoing packets
  - .1.3.6.1.4.1.10812.1.2.2.1.8 – the number of passed encrypted incoming packets
  - .1.3.6.1.4.1.10812.1.2.2.1.9 – the number of blocked encrypted incoming packets

- .1.3.6.1.4.1.10812.1.2.2.10.10 – the number of passed encrypted outgoing packets
- .1.3.6.1.4.1.10812.1.2.2.1.11 – the number of blocked encrypted outgoing packets

ViPNet Linux uses the following SNMP Traps:

- .1.3.6.1.4.1.10812.1.3.1 – the ViPNet control program is started
- .1.3.6.1.4.1.10812.1.3.2 – the ViPNet control program is stopped
- .1.3.6.1.4.1.10812.1.3.3 – the MFTP daemon is started
- .1.3.6.1.4.1.10812.1.3.4 – the MFTP daemon is stopped
- .1.3.6.1.4.1.10812.1.3.7 – the failoverdis started (only in the active mode)
- .1.3.6.1.4.1.10812.1.3.8 – the failoverd is stopped (only in the active mode)
- .1.3.6.1.4.1.10812.1.3.9 – the failoverd is switched from the passive into the active mode)

As a rule, the NMS allows you to configure a certain respond to each trap (sending an e-mail message, sms, etc.)

12 Checking Dumps of Abnormal Termination of ViPNet Linux Programs

All Linux operating systems give an opportunity to create programs' dumps at their abnormal termination. As a rule, the abnormal termination occurs quite rarely, so if such a situation occurs you should send dumps, saved by the system, to the program developers. These dumps contain important information that is necessary to find and eliminate causes of errors, and to increase a reliability and a stability of ViPNet software functioning as well.

Starting from version 2.8-284, the programs, included in the ViPNet Linux software, create the dumps automatically and check a presence of the dumps.

The programs, for which such dumps will be created and checked, are listed below:

- iplircfg
- dbviewer
- iplir_remote_info
- iplir-config-manager
- failoverd
- iplirpasswd
- mftpd
- mailtrans
- mftp_remote_info

Coredumps folder will be created in the key databases folder. Inside the coredumps folder each ViPNet program creates a subfolder with a name, corresponding to a name of the ViPNet program. When such a program starts, a presence of a corresponding subfolder will be checked and if there is no such a subfolder, it will be created. After that, a presence of dumps for all programs is checked and as soon as such dumps are found, a corresponding message will be sent to terminal with a request to send these dumps to Infotecs with specifying ViPNet version number. Also this message will be written in syslog. As soon as the dumps are sent you can delete these dumps from your disk to set disk space free.

To create dumps, a free disk space in the key databases folder is required. If the free disk space is insufficient, the dumps won't be created. To avoid such situations a check of free space will be done at program start. If a free disk space is insufficient (see chapter 2), a message will be sent to syslog (and to terminal, if it is a program start). Also a check of free disk space is being done every five minutes during working control daemon (iplircfg).

13 ViPNet Linux Troubleshooting Logs

The services (daemons) included in the ViPNet software always create logs allowing to keep tracks of all activities of the specified service and to troubleshoot any errors or malfunctions occurred during this service operation.

These logs (especially when the highest logging level is set) contain detailed information about all internal processes occurring deep in the ViPNet daemons and are indispensible for developers.

This information along with abnormal termination dumps is used to research and troubleshoot ViPNet software-related issues.

Logs can be recorded and saved:

- as a text file in a directory defined by the settings (by default).
- in the general Linux OS log file by a special service – syslogd daemon.

Logging is configured by several parameters defined in the [debug] section of a configuration file of a corresponding ViPNet Linux service. The following parameters are defined in the [debug] section:

- log location identifier – is defined by the debuglogfile parameter. This identifier has the following format: **<protocol specifier>:<URL specifier for this protocol>.** The current version, as mentioned above, supports two kinds of logging protocols:
  - Logging into a file. In this case the protocol specifier should have the file value, and the URL specifier should contain the pathname of the log file, e.g.:

    debuglogfile= file:/var/log/iplircfg.debug

    Set up as this, the log for the control daemon will be recorded in a specified file, like in the previous versions of ViPNet Linux software.
  - Logging into the Linux OS system log. In this case the protocol specifier should have the syslog value, and the URL specifier should consist of two parts divided by a dot and defining the source (facility) and the importance (level) for the system log messages, e.g.:

    debuglogfile= syslog:daemon.debug

    The source of messages (facility) defines the process of generating messages. The source of messages can have the following values:
    - kern (kernel);
    - user (user programs);
    - mail (mail system);
    - daemon (daemons).

    The importance of messages (level) defines if the messages are critical for Linux OS administrator and can have the following values:
    - err (error);
    - info (information message);
    - debug (debug message).

    If the syslog protocol specifier with defined facility and level is used, log messages will be sent to the syslogd system daemon that will process them according to its settings and save to the system log. Besides saving messages in system log local files, there is also a possibility to configure the syslogd daemon so that it would send the messages to a remote server. You can find more information on the functionality and configuring the

syslogd daemon in a corresponding syslogd help guide from a Linux distributive delivery set.

- logging level – defined in the debuglevel parameter. Logs are text files, located in the folder /var/log by default. Logs location and names are specified in configuration files of corresponding daemons (see section 6.2). Logging can be done at different levels. Logging level defines an amount of information that will be written into the log. There are five logging levels, numbering from 1 up to 5. The higher a logging level is, the more information the log contains. The logging level of each daemon is specified in its configuration file (see section 6.2). The default logging level is 3, it corresponds with the middle detail level. In the most cases this level provides sufficient information for daemon work diagnostics. However, in some cases, more detailed data can be required and you need to set a higher logging level. In this case it's recommended to address to support service of the Infotecs to properly choose a logging level necessary for error research and recovery.

As mentioned above, the higher a logging level is, the more information the log contains, and, correspondingly, the log file size increases. If you use the default (3) logging level, the log file size increases by 150-300 kilobytes in an hour depending on traffic (information density), number of workstations etc. At the higher logging level the log file size increases faster. If the saving of service ViPNet logs in the system log is properly set, the excess of the log file maximum size is unlikely, as rotation of the system log files is set in Linux OS by default. But saving ViPNet service logs in local files with the file protocol specifier in the debuglogfile parameter may cause such problems.

To prevent an increase of log file size and a critical decrease of free disk space, ViPNet Linux software uses logs rotation and archiving. To meet this goal, ViPNet Linux uses a system utility logrotate, which is a standard tool for logs rotation in Linux OS.

Attention! The logrotate utility and the cron scheduler are must-haves for ViPNet logging system to work correctly.

When installing ViPNet Linux software, a vipnet file will be created in /etc/logrotate.d folder. In this file rotation settings are defined in /var/log: iplircfg.debug.log, failover.debug.log, mftp.debug.log. The default parameter settings are listed below:

- execute daily rotation;
- store logs for the last 7 days;
- compress old logs;
- call corresponding script with parameter logrotate after rotation and before compression (option postrotate):

  for iplircfg.debug.log – script iplir

  for failover.debug.log – script failover

  for mftp.debug.log – script mftp

Attention! Absolute paths to log files of the corresponding ViPNet daemons are used in this file. It's not recommended to change these paths. However, if you have decided to change parameter debuglogfile (see section 6.2), it's necessary to change it in configuration file /etc/logrotate.d/vipnet as well. Also, it's impossible to move log files to another location by changing them on symbolic links.

Appendix A. Daemons included in the ViPNet Linux software

When ViPNet Linux is running, there are some daemon programs permanently present in the memory. Below you can see the list of them with a brief description of their purpose.

- iplircfg – the main control daemon in ViPNet implementing the features of the IP Addresses Server, keeping the log, etc. It is started with the command iplir start and stopped with the command iplir stop.
- mftpd – the MFTP daemon processing and forwarding business messages, receiving the updates of host links and key sets, etc. It is started with the command mftp start and stopped with the command mftp stop.
- failoverd – the daemon of the failover system (see its description in 9 section). It is started with the command failover start or failover start-active and stopped with the command failover stop.

All daemon programs included in the ViPNet Linux SW operate as operating system processes, and may be viewed by running, e.g., the ps aux or the top command. Here the name of the corresponding process shows its current state, i.e., the functions currently performed by it. The beginning of the process header contains the daemon name (iplircfg, mftpd, failoverd), followed a colon and a space, and then by a string identifying its current state.

Below is the lost of possible states for the ViPNet Linux SW daemon processes along with brief descriptions.

The iplircfg control daemon:

- initializing – this state means that the daemon initialization procedure is being performed, which is executed before performing its main functions. Usually, the initialization procedure includes reading and analyzing the configuration, loading information into the driver, etc.m and takes a short time (from one to several tens of seconds).
- working – this state means that the initialization procedure has completed successfully, and the daemon entered the state of performing its main functions. This state remains until the control daemon process operation is terminated.

In case of the operation of the failover system in the hot server sparing cluster mode (see the "The ViPNet Linux SW administrator guide to the failover system"), when the control daemon is started in the passive mode, the end of the process header is appended with the following mode notation: (passive).

The mftpd daemon:

- initializing – this state means that the daemon initialization procedure is being performed, which is executed before performing its main functions. Usually, the initialization procedure takes a short time (from one to several tens of seconds).
- updating – means that the remote key, reference table or SW update process is started.
- transferring – means that the active envelope transfer over one of the MFTP links (except SMTP/POP3) is currently performed, both in case of data reception and data transmission.
- connecting - means that attempts to connect to one of the remote nodes (except the SMTP/POP3 link), or to the mftpd on the passive part are being made, in case of

operation as a part of the hot server sparing cluster (see the "ViPNet Linux SW administrator guide to the failover system").

- idle – means that the mftpd daemon is currently in the idle mode, i.e., performs not the main functions, but some functions unrelated to any of the above states.

In any state the mftpd process header is also appended with the failover system operating mode:

- (single) – a standalone operating mode;
- (active) – an active operating mode as a part of the hot sparing cluster;
- (passive) – a passive operating mode as a part of the hot sparing cluster.

The failoverd failover system daemon:

- initializing – this state means that the daemon initialization procedure is being performed, which is executed before performing its main functions. Usually, the initialization procedure includes reading and analyzing the configuration, configuring network interface parameters, restarting other daemons, etc., depending on the failover system operating mode. The initialization procedure takes a short time (from one to several tens of seconds).
- working – this state means that the initialization procedure has completed successfully, and the daemon entered the state of performing its main functions.
- switching to active – this means that the configuration of the system when switching from the passive to the active mode is being performed. This state is only possible when the failover system operates in the hot sparing cluster mode.
- rebooting – means that the failoverd daemon has initiated a system reboot

In any state (except switching to active) the process header is appended with the failover system operating state:

- (single) – a standalone operating mode;
- (cluster) (active) – an active operating mode as a part of the hot sparing cluster;
- (cluster) (passive) – a passive operating mode as a part of the hot sparing cluster.

Appendix B. ViPNet Linux kernel modules

When ViPNet Linux is running, there are some kernel modules permanently present in the memory (they are also called drivers). Below you can see the list of them with a brief description of their purpose.

- drviplir – the main ViPNet driver processing network packets. Produces a kernel thread visible in the list of processes as [kiplird], kills it when unloaded. Loaded with the command iplir start, unloaded with the command iplir unload. Cannot be unloaded manually with the command rmmod. For the correct operation, it must be loaded after the itcscrpt driver. For the correct operation with the watchdog system, it must be loaded after the itcswd driver.
- itcscrpt – the cryptographic driver, performing cryptographic operations by request from the drviplir driver. Loaded using the iplir start command, and unloaded using the iplir unload command.

- itcswd – the driver of the failover system which controls the system state. Produces a kernel thread visible in the list of processes as [kwatchdogd], kills it when unloaded. Loaded with the command vipmod start, can be unloaded with the command rmmod after the failoverd is stopped and the drviplir is unloaded.

Only an experienced Linux administrator should manually unload and load ViPNet drivers.

Appendix C. Changes made in the system when ViPNet Linux is installed and uninstalled

The ViPNet Linux distribution package can be supplied in two variants: dynamic and static. In the dynamic variant the common parts of code in various programs are supplied as dynamic link libraries. It saves disk space and RAM, but dynamic link libraries may fail to work if the system is set up in a nonstandard way. In the static distribution mode all programs are independent and do not require any dynamic link libraries. The static distribution package has the suffix –static in its name. Further in this appendix, the behavior of both distribution packages is described unless stated otherwise.

Installation:

1. The following files are saved to the /sbin directory:

   iplircfg
   iplircfg.crg
   dbviewer
   dbviewer.crg
   iplir_remote_info
   iplir_remote_info.crg
   iplir-config-manager
   iplir-config-manager.crg
   failoverd
   failoverd.crg
   failover_info
   failoverd_info.crg
   unmerge
   fetchmail
   fetchmail.crg
   iplirpasswd
   iplirpasswd.crg
   mftpd
   mftpd.crg
   mailtrans
   mailtrans.crg
   mftp_remote_info
   mftp_remote_info.crg
   iplir_monitoring_export
   iplir_monitoring_export.crg
   monitoringcfg
   monitoringcfg.crg
   iplir
   vipmod
   failover

mftp
rmiplir

2. The following files are saved to the /usr/bin directory:
   lha
   lha.crg

3. The following files are saved to the /etc directory:
   iplirprg
   iplirprg.crg
   iplirpsw
   iplirnetpsw
   failover.ini
   iplir.serv
   iplirSKnums
   /etc/logrotate.d/vipnet

4. The directory /var/spool/vipnet is created.
5. The following directories are searched for in the /etc/rc.d directory first and then, if they are not there, in /etc:
   init.d
   rc0.d
   rc1.d
   rc2.d
   rc3.d
   rc4.d
   rc5.d
   rc6.d

   The following symbolic links are placed in the init.d directory:
   vipmod -> /sbin/vipmod
   failover -> /sbin/failover
   Also the watchdog script is written in this directory for the compatibility with earlier versions.
   The following symbolic links are placed in the rc0.d, rc1.d, rc2.d directories:
   K<XX> failover -> /etc/init.d/failover
   K<YY> vipmod -> /etc/init.d/vipmod
   where <XX>, <YY> may change from version to version.

   The following symbolic links are placed in the rc2.d, rc2.d, rc4.d, rc5.d directories:
   S<XX> vipmod -> /etc/init.d/vipmod
   S<YY> failover -> /etc/init.d/failover
   where <XX>, <YY> may change from version to version.

6. The following files are saved to the directory /lib/modules/<kernel_version>/misc:

   drviplir.o
   itcswd.o
   itcscrpt.o
   <kernel_version> is determined as a result of executing the command uname –r.

7. If you use the dynamic distribution package, the directory /usr/lib/vipnet is created and library files the list of which may vary from version to version is saved there.

The following files from the /etc directory are NOT overwritten when a new version is installed over the old one (except the cases of installation without backup of the current ViPNet configuration):

iplirpsw
iplirnetpsw
failover.ini

All other files are overwritten. If the static distribution package is installed over the dynamic one, the directory /usr/lib/vipnet is not deleted, though it is no longer used.

When ViPNet is removed from the system, all enumerated files and links are deleted (including /usr/lib/vipnet even if the static version is removed), except the file /sbin/rmiplir.

Appendix D. Peculiarities of working with ViPNet Linux on various hardware configurations

This appendix contains the information about the peculiarities of the work of ViPNet Linux with certain hardware (for example, on some brand servers) and about settings that should be specified in each case.

Supermicro Servers: SuperServer 5013G-i (SYS-5013-Gi)

You should disable the support of Advanced Power Management (APM) on these servers for ViPNet Linux to work on them properly. You can do it either in the process of building the kernel by disabling the corresponding option or when the system is being loaded by specifying the parameter apm=off. You should use ACPI in case you still need power management. This interface supports all features provided by APM and works well on such servers.

Ethernet network adapters with support of “Scatter/Gather I/O”and TSO technologies

To increase network packets transmission rate, “Scatter/Gather I/O” and “TSO (TCP Segmentation Offload) technologies are used in modern Ethernet adapters by default. These technologies allow to optimize receiving/passing packets by means of adapter hardware. These technologies are not compatible with working of current version of IpLir driver and must be switched off for correct work of ViPNet software.

To switch off a support of these above technologies you can use prevalent utility ethtool:

ethtool –K eth<N> sg off,

ethtool –K eth<N> tso off

where <N> - network interface number.

To check a status of “Scatter/Gather I/O” and “TSO” you can use the following command:

ethtool –k eth<N>,

where <N> - network interface number.

If these mechanisms are switched off, the command will output the following message:

scatter-gather: off

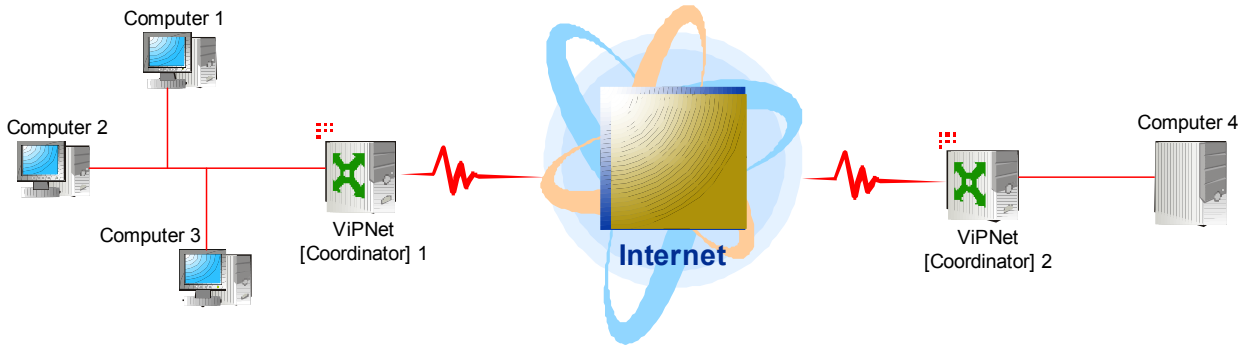
tcp segmentation offload: off

For more information about the utility ethtool, included in your Linux delivery set, you can address to manual (man ethtool).

When the ViPNet SW is started during OS startup, the specified optimization technologies are automatically disabled, with the corresponding notification output into syslog.

Appendix E. An example of configuring tunnels using the ViPNet Coordinator.

Let us take one of the commonly used schemes of using tunnels in ViPNet. Suppose there are two offices connected via the Internet. In one of the offices there is a server and computers from the other office connect to it. You need to encrypt the entire traffic in the process of data exchange via the Internet. At the same time, it is impossible or undesirable for some reasons to install ViPNet directly on the computers participating in the information exchange. In order to solve this task, a ViPNet Coordinator is installed in each of the offices. The scheme looks the way it is shown in the picture.



Each coordinator has two network interfaces one of which has a real address and is connected to the Internet, while the other one has a private address and is connected to the office LAN. Let Coordinator 1 have external address 195.210.139.22 and internal address 192.168.1.1, computers connected to it (1, 2, 3) have addresses from 192.168.1.2 to 192.168.1.4, Coordinator 2 has external address 194.87.0.8 and internal address 192.168.2.1, while the computer connected to it (4) has address 192.168.2.2. In order to configure the correct work of tunnels, you should specify the following settings in the file iplir.conf on the Coordinators:

- Coordinator 1:
  - Insert the following line in its own [id] section:
    tunnel= 192.168.1.2-192.168.1.4 to 192.168.1.2-192.168.1.4
  - Insert the following line in the [id] section of the Coordinator 2:
    tunnel= 192.168.2.2-192.168.2.2 to 192.168.2.2-192.168.2.2
- Coordinator 2:
  - Insert the following line in its own [id] section:
    tunnel= 192.168.2.2-192.168.2.2 to 192.168.2.2-192.168.2.2
  - Insert the following line in the [id] section of the Coordinator 1:
    tunnel= 192.168.1.2-192.168.1.4 to 192.168.1.2-192.168.1.4

The internal interfaces of both coordinators must not work in mode 1. If they work in mode 2 or 3, you should specify permitting filters for tunneled workstations in file firewall.conf (section. 6.3).:

For Coordinator 1 in table [local] of firewall.conf file it is necessary to set the following rules

rule= proto any from 192.168.1.2-192.168.1.4

to 192.168.2.2 pass
rule= proto any from 192.168.2.2 to 192.168.1.2-192.168.1.4 pass

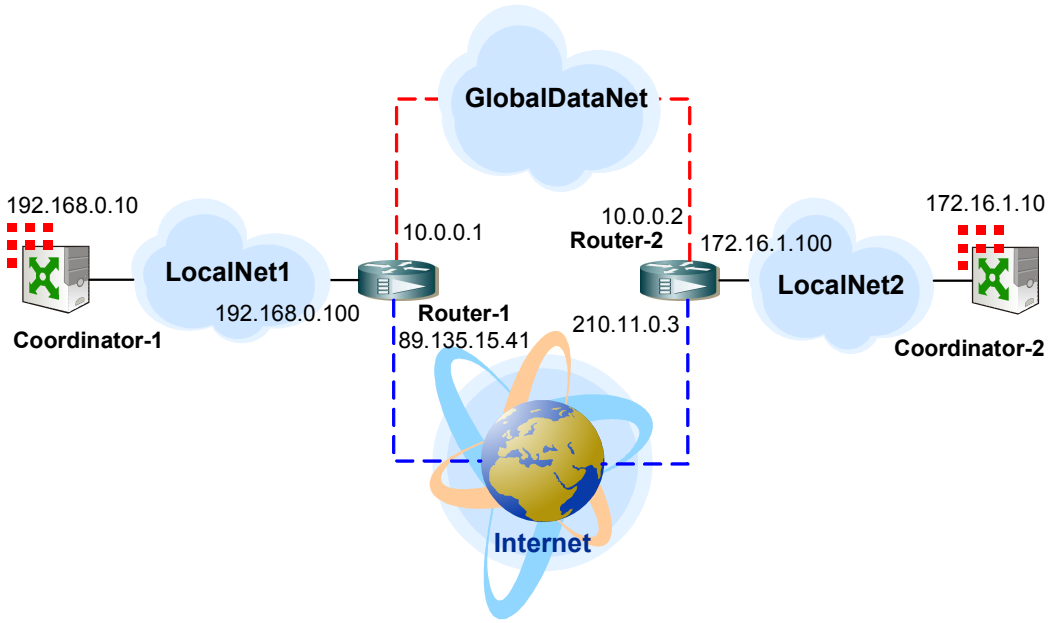
The same rules should be set for Coordinator 2 in the table [local] of the file firewall.conf:
rule= proto any from 192.168.1.2-192.168.1.4 to 192.168.2.2 pass
rule= proto any from 192.168.2.2 to 192.168.1.2-192.168.1.4 pass

If the internal interface of some Coordinator works in mode 4 or 5, there is no need to configure filters.

After the ViPNet daemon with the specified settings is started, computers 1, 2 and 3 will be able to address computer 4 using address 192.168.2.2 and packets will be encrypted between Coordinator 1 and Coordinator 2.

Appendix F. Examples of configuring the work of vipnet coordinators via fixed alternative channels

Here is a pilot model of information exchange between two ViPNet coordinators having two alternative access channels apparently named **GlobalDataNet** and **Internet**.



Both coordinators can exchange information unlimited via any of the channels concerned. It is implied that the nodes **Router-1** and **Router-2** route the packets for these channels. During it the source address will be changed for the packets routed in the Internet and forwandig will be set up for the address of the corresponding coordinator (static address translation) for incoming from the Internet packets with destination port 55777. The gateway of each coordinator is installed by default on the internal interface of the corresponding router. “Static NAT” mode is set on both coordinators (see 6.5.3).

Below you can find several solutions of the typical problems of managing the coordinators exchange channels.

1. Choosing GlobalDataNet as a fixed coordination channel.

First of all, create a [channels] section in iplir.conf file for each coordinator and define two alternative coordination channels correlating them with the corresponding groups:

[channels]

channel= GlobalDataNet, DataNetGroup

channel= Internet, InternetGroup

Then correlate the coordinators to the corresponding group and set access parameters for each coordinator. To do this, define the following settings in iplir.conf file on **Coordinator-1** in the [id] section for **Coordinator-2**:

group= DataNetGroup

channelfirewallip= GlobalDataNet, 10.0.0.1

channelfirewallip= Internet, 89.135.15.41

After the control daemon's startup with these settings both coordinators will exchange information only via the **GlobalDataNet** channel.

2. Switching to the Internet channel.

To make the coordinators to exchange information via the **Internet** channel, do the following. Change the group parameter value in iplir.conf file on **Coordinator-1** in the [id] section for **Coordinator-2:**

group= Internet

Make the same changes on **Coordinator-2** in the [id] section for **Coordinator-1**. After the control daemon's startup with these settings both coordinators will exchange information only via the **Internet** channel.

3. Finishing the work via fixed channels.

To stop working via fixed channels, in this example, you should enter iplir.conf file on each coordinator in the [channels] section and cancel groups registration in the channels defined by the channel parameters:

[channels]

channel= GlobalDataNet

channel= Internet